University of California
Santa Barbara

Coordination Strategies for Human Supervisory Control of Robotic Teams

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Mechanical Engineering

by

Jeffrey Russell Peters

Committee in charge:

Professor Francesco Bullo, Chair
Professor Brad E. Paden
Professor Jeff Moehlis
Professor Yasamin Mostofi

June 2017

The Dissertation of Jeffrey Russell Peters is approved.

Professor Brad E. Paden

Professor Jeff Moehlis

Professor Yasamin Mostofi

Professor Francesco Bullo, Committee Chair

June 2017

Coordination Strategies for Human Supervisory Control of Robotic Teams

Jeffrey Russell Peters

To everyone who has given me words of encouragement.

Acknowledgements

A special thank you goes to my advisor, Francesco Bullo, for the crucial role that he has played in my professional development. Dr. Bullo has been nothing but supportive of me, encouraging me to pursue my ideas and to get involved with various endeavors that have made me into the scientist that I am today. He has shown incredible patience and persistence in helping me to gain the confidence that has allowed me to reach my goals. I will always be grateful to him for his efforts and I am sure the lessons that he has taught me will serve me well throughout my career.

In addition to Dr. Bullo, I have been extremely fortunate to have collaborated with a number of great scientists over the years, and I am thankful to each of them for the roles that they have played in my career. In particular, I would like to express my deepest gratitude to Amit Surana from United Technologies Research Center (UTRC). I was lucky enough to work with Dr. Surana on the collaborative project that comprised the bulk of my Ph.D work, and his mentorship has been crucial to the project's success and my own scientific maturation. A special thanks also goes to Luca Bertuccelli for his help during my internships at UTRC, and for all of the professional advice that he has given me. Also deserving of my gratitude are all of the fantastic people at UTRC, including (but not limited to) Michael Giering, Tong Sun, Bob Labarre, Isaac Cohen, Andrej Banazjuk, Paul O'Neil, Billy Sisson, Alex Shilov, and Alex Dorgan.

I also extend my thanks to the rest of my collaborators from the ICB SCORCH project that have not yet been acknowledged, namely (i) Grant Taylor and Terry Turpin from the US Army Aviation Development Directorate, (ii) Miguel Eckstein and Arturo Deza from the UCSB psychology department, and (iii) all of the people at ICB that made the project a reality.

Next, I would like to thank Fabio Pasqualleti, Vhaibhav Srivastava, John Simpson,

Rush Patel, David Copp, Tie Bo Wu, and Dan Wilson who all helped me to establish myself as a graduate student and who continued collaboration with me in various forms throughout the years. Thank you also to Mishel George, Pushkharini Agharkar, Wenjun Mei, and all of the other present and past students from my lab group, for the many stimulating discussions, both inside and outside of the lab, that have made my stay at UCSB truly enjoyable. Further, I would like to say thank you to my PhD committee members, Jeff Moehlis, Brad Paden, and Yasamin Mostofi, for their efforts and encouragement, as well as their help with the PhD process. A thank you also goes out to Sean Wang, and all of the other students that I have mentored and taught. I have learned as much from them as they have learned from me, and it has been a pleasure to witness their continued successes.

Lastly, I would like to thank all of my friends and family for their support. Most notably, the biggest thank you of them all goes to my wife Denisse, who has supported me through all of the ups and downs that life has brought.

Curriculum Vitæ

Jeffrey Russell Peters

Education

2017	**Ph.D. Mechanical Engineering** University of California Santa Barbara, CA, USA.
2015	**M.A. Applied Mathematics** University of California Santa Barbara, CA, USA
2013	**M.S. Mechanical Engineering** University of California Santa Barbara, CA, USA
2011	**B.S. Mechanical Engineering** University of Illinois Urbana - Champaign, Illinois, USA

Research Experience

2011 - 2017	**Graduate Student Researcher** University of California, Santa Barbara Investigated problems related to human supervisory control and multi-agent coordination, including (i) routing and load-balancing strategies for multi-agent surveillance missions, (ii) human behavioral modeling and decision support development using physiological sensing, (iii) robust scheduling strategies for sequential task-analysis, (iv) joint routing/scheduling for supervisory missions, (v) sensor network localization under cyclic constraints, and (vi) optimal coordination of active cameras for smart intruder detection.
Summer 2014/15	**Systems Department Intern** United Technologies Research Center Performed simulated and experimental studies in order to model human operator behavior and increase performance. In particular, used eye-tracking to assess usability of a supervisory interface, and developed robust scheduling strategies for multi-operator missions.

Teaching Experience

2017	**Certificate in College and University Teaching (CCUT)** University of California, Santa Barbara

Spring 2017	**Teaching Associate** Course: ME 179P, *Intro to Robotic Planning and Kinematics* Mechanical Engineering, University of California, Santa Barbara
Summer 2016	**Teaching Associate** Course: ME 16, *Engineering Mechanics: Dynamics* Mechanical Engineering, University of California, Santa Barbara
Fall 2015	**Teaching Assistant** Course: ME 104, *Mechatronics* Mechanical Engineering, University of California, Santa Barbara
Spring 2014	**Teaching Assistant** Course: ME 16, *Engineering Mechanics: Dynamics* Mechanical Engineering, University of California, Santa Barbara
Fall/Winter 2013	**Instructor** Course: *Thinking Robotics: Teaching Robots to Make Decisions* School for Scientific Thought, University of California, Santa Barbara
Fall 2011	**Teaching Assistant** Course: ME 104, *Mechatronics* Mechanical Engineering, University of California, Santa Barbara

Mentoring

2016-2017	**Jake Carrade, Alan Cao, Sean Wang, Viswa Rao, and Landon Peik** Program: Mechanical Engineering Capstone Design Team University of California, Santa Barbara
2016-2017	**Franklin Zheng** Program: B.S./M.S. Research University of California, Santa Barbara
2016-2017	**Sean Wang** Program: Undergraduate Research University of California, Santa Barbara
Spring 2016	**Tirion Wray** Program: Undergraduate Research University of California, Santa Barbara
Summer 2014	**Ariana Del Toro** Program: RISE Summer Internship University of California, Santa Barbara
Summer 2014	**Heather Vermilyea** Program: School for Scientific Thought Course Development University of California, Santa Barbara

Professional Service

Technical Reviewer

Journal	IEEE Transactions on Control of Network Systems
	IEEE Transactions on Control Systems Technology
	EEE Transactions on Human Machine Systems
	Automatica
Conference	American Control Conference
	IFAC World Congress

Publications

Journal Articles

1. J.R. Peters, A. Surana, G.S. Taylor, T. Turpin, and F. Bullo. *UAV Surveillance Under Visibility and Dwell-Time Constraints* (2017). Submitted.
2. J.R. Peters, A. Surana, and F. Bullo. *Joint Scheduling and Routing for Collaborative Human-UAV Persistent Surveillance Missions* (2017). Submitted.
3. J.R. Peters, S.J. Wang, A. Surana, and F. Bullo. *Cloud-Supported Coverage Control for Persistent Surveillance Missions*, *ASME Journal of Dynamic Systems, Measurement, and Control* **139** (2017), no. 8 081011 - 081011-12.
4. J.R. Peters and L. Bertuccelli. *Robust Task Scheduling for Multi-Operator Supervisory Control Missions*, *AIAA Journal of Aerospace Information Systems* **13** (2016), no. 13 393 - 406.
5. J.R. Peters, D. Borra, B.E. Paden, and F. Bullo. *Sensor Network Localization on the Group of 3D Displacements*, *SIAM Journal on Control and Optimization* **53** (2015), no. 6 3534 - 3561.
6. J.R. Peters, V. Srivastava, G.S. Taylor, A. Surana, M.P. Eckstein, and F. Bullo. *Human Supervisory Control of Robotic Teams: Integrating Cognitive Modeling with Engineering Design*, *IEEE Control Systems Magazine* **35** (2015), no. 6 57 - 80.
7. F. Pasqualetti, F. Zanella, J.R. Peters, M. Spindler, R. Carli, and F. Bullo. *Camera Network Coordination for Intruder Detection*, *IEEE Transactions on Control Systems Technology* **22** (2013), no. 5 1669 - 1683.

Conference Articles

1. A. Deza, J.R. Peters, A. Surana, G. S. Taylor, and M. Eckstein. *Attention Allocation Aid for Visual Search*, in *CHI Conference on Human Factors in Computing Systems*, (Denver, CO, USA), pp. 220 - 231, May, 2017.
2. J.R. Peters, S.J. Wang, and F. Bullo. *Coverage Control with Anytime Updates for Persistent Surveillance Missions*, in *American Control Conference* (Seattle, WA, USA), pp. 265 - 270, May, 2017.

3. J.R. Peters and L. Bertuccelli. *Robust Scheduling Strategies for Collaborative Human-UAV Missions*, in *American Control Conference*, (Boston, MA, USA), pp. 5255 - 5262, June, 2016.

Software

1. J.R. Peters and R. Patel. *Thinking Robotics: Teaching Robots to Make Decisions*. http://www.teachengineering.org, 2015.

Books/Teaching Curriculum

1. J.R. Peters and Contributors. *The AreaCon Library*. http://www.areacon.org, 2016.

Thesis

1. J.R. Peters. *Camera Coordination for Intruder Detection in 1D Environments*. MS Thesis, Mechanical Engineering Department, University of California at Santa Barbara, December 2014.

Misc./Unpublished Work

1. J.R. Peters, A. Surana, and Luca Bertuccelli. *Eye-Tracking Metrics for Task-Based Supervisory Control*. arXiv preprint arXiv:1506.01976, 2015.

Abstract

Coordination Strategies for Human Supervisory Control of Robotic Teams

by

Jeffrey Russell Peters

Autonomous mobile sensor teams are crucial to many civilian and military applications. These robotic teams often operate within a larger *supervisory system*, involving human operators who oversee the mission and analyze sensory data. Here, both the human and the robotic system sub-components, as well as interactions between them, must be carefully considered in designing effective mission coordination strategies.

This dissertation explores a series of representative sub-problems relating to the analysis and coordination of both mobile sensors and human operators within supervisory systems. The content herein is presented in three parts: Part I focuses on coordinating operator behavior independently (operator-focused methods), Part II focuses on coordinating mobile-sensor behavior independently (sensor-focused methods), and Part III focuses on jointly coordinating both operator and mobile sensor behavior (joint methods). The content herein is primarily motivated by a particular application in which Unmanned Aerial Vehicles collect visual imagery to be analyzed by a remotely located operator, although many of the results apply to any system of similar architecture.

Specifically, with regard to operator-focused methods, Chapter 2 illustrates how physiological sensing, namely eye tracking, may provide aid in modeling operator behavior and assessing the usability of user interfaces. The results of a pilot usability study in which human observers interact with a supervisory control interface are presented, and eye-tracking data is correlated with various usability metrics. Chapter 3 develops robust scheduling algorithms for determining the ordering in which operators should pro-

cess sensory tasks to both boost performance and decrease variance. A scenario-based, Mixed-Integer Linear Program (MILP) framework is presented, and is assessed in a series of numerical studies.

With regard to sensor-focused methods, Chapters 4 and 5 consider two types of supervisory surveillance missions: Chapter 4 develops a *cloud-based* coverage strategy for persistent surveillance of planar regions. The scheme operates in a dynamic environment, only requiring sporadic, unplanned data exchanges between a central cloud and the sensors in the field. The framework is shown to provide collision avoidance and, in certain cases, produce convergence to a Pareto-optimal coverage configuration. In chapter 5, a heuristic routing scheme is discussed to produce Dubins tours for persistent surveillance of discrete targets, each with associated visibility and dwell-time constraints. Under some assumptions, the problem is posed as a constrained optimization that seeks a minimum-length tour, while simultaneously constraining the time required to reach the first target. A sampling-based scheme is used to approximate solutions to the constrained optimization. This approach is also shown to have desirable resolution completeness properties.

Finally, Chapter 6 explores joint methods for coordinating both operator and sensor behavior in the context of a discrete surveillance mission (similar to that of Chapter 5), in which UAVs collect imagery of static targets to be analyzed by the human operator. In particular, a method is proposed to simultaneously construct UAV routes and operator schedules, with the goal of maintaining the operator's task load within a high-performance regime and preventing unnecessary UAV loitering. The full routing/scheduling problem is posed as a mixed-integer (non-linear) program, which can be equivalently represented as a MILP through the addition of auxiliary variables. For scalability, a MILP-based receding-horizon method is proposed to incrementally construct suboptimal solutions to the full optimization problem, which can be extended using a scenario-based approach (similar to that of Chapter 3) to incorporate robustness to operator uncertainty.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of mobile sensors is becoming increasingly common in both civilian and military applications, since autonomous agents can provide support in tasks that are too dangerous, too expensive, or simply too difficult for humans to perform unaided. Example applications that can benefit from autonomous sensors include search and rescue, forest fire or oil spill monitoring, surveillance and reconnaissance, transportation and logistics, and hazardous waste cleanup [2, 3, 4].

Naturally, applications involving autonomous sensors require intelligent and practical strategies to govern sensor behavior in the presence of numerous constraints that arise in operational scenarios. However, these sensor coordination strategies alone may not be enough to guarantee effective operation of the overall system. Indeed, in many realistic missions, mobile sensors are only a part of a much larger system involving diverse data sources and analysis tools. These complex systems often contain both human and robotic elements and, in many cases, system functionality relies on the human operator's ability to process information generated by their autonomous partners quickly and accurately [5, 6]. The incredible amount of data generated by modern sensors makes human operators susceptible to *information overload*, which can have detrimental effects on performance

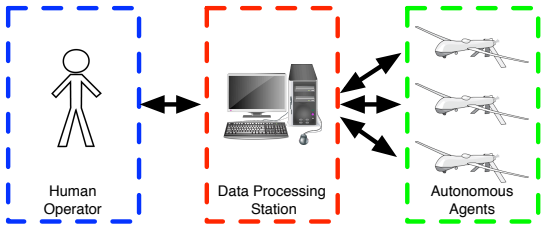

Figure 1.1: A typical human supervisory control setup consisting of 3 main components: the human operator(s), the data-processing station, and the autonomous agents. Human operator(s) interact with the autonomous agents through the data-processing station. The degree to which human performance and input affects automation, as well as the method by which sensor data is presented to the operators is determined by the data-processing station and its internal functionalities.

and may lead to dire consequences [7]. Therefore, it is not only necessary to coordinate the mobile sensors, but also to understand human operator behavior and to properly mediate their relationship with autonomous agents. This "systems engineering" perspective has been the topic of many recent research initiatives like the National Robotic Initiative [8], which emphasizes collaboration between humans and their robotic partners and envisions symbiotic mechanisms to facilitate interactions between diverse system components.

In this dissertation, we explore various mechanisms for both understanding and coordinating different aspects of a particular type of multi-agent system in which human operators are responsible for overseeing autonomous agents and providing feedback based on sensor data. In the control systems community, the term *human supervisory system* (or simply *supervisory system*) is often used as a shorthand reference for systems with this type of architecture [9, 10, 11]. In a typical human supervisory control application, the operator indirectly interacts with autonomous agents via a central data-processing station (Figure 1.1). As such, system designers not only can develop sophisticated algorithms for guiding autonomous agents, but also have the opportunity to easily incorporate automated utilities to control how information is presented to the operator, and how the input provided by the operator is used by automated systems. The goal of these utilities is to take advantage of the inherent robustness and adaptability of human operators,

while mitigating adverse effects such as unpredictability and performance variability. In some contexts, to meet the goal of single-operator supervision of multiple automated sensor systems, such facilitating mechanisms are not only useful, but necessary for practical use [12, 13]. Although many of the topics discussed herein are applicable to a wide-range of supervisory systems, our analysis is motivated primarily by a particular supervisory application in which sensory data regarding some target or key event is collected by a team of Unmanned Aerial Vehicles (UAVs) and subsequently analyzed by a remotely located human operator (within either a helicopter or ground control station). As such, most of the topics discussed herein are placed within this context.

Our discussion is divided into three parts, each of which presenting subproblems that illustrate a particular approach supervisory system design: In Part I, we focus on methods that seek to optimize system performance by improving the operator experience and moderating operator behavior. In Part II, we focus on the design of multi-agent coordination algorithms for the mobile sensors, in order to better divide the sensing workload and plan more efficient routes. Finally, in Part III, we focus on methods that seek to improve overall system performance by jointly optimizing over the operator schedule and the autonomous vehicle routes.

1.1 Subproblem Descriptions

Each chapter of this dissertation addresses a different subproblem relating to the design of an effective supervisory control system for surveillance missions involving UAVs. A brief description and motivation for each subproblem is provided here.

Part I: Operator-Focused Methods

Human Modeling and Usability Analysis Using Physiological Sensing: An understanding of operator behavior is a crucial component to any supervisory coordination scheme. One set of tools that can aid in developing this understanding includes physiological sensors such as eye-trackers. Eye-tracking has become a common means of analyzing operator interactions with interfaces [14], and is widely used in psychology [15], communications [16], and more recently, engineering [17].

In Chapter 2, we investigate the potential benefit of using eye-tracking to understand operator behavior and to assess interface usability in the context of supervisory control. We present a brief, pilot usability study in which gaze information was recorded while human subjects interacted with a particular supervisory control interface called the Research Environment for Supervisory Control of Heterogenous Unmanned Vehicles (RESCHU) [18]. In this short study, we demonstrate how metrics that can provide insight into "typical use" of a given supervisory interface can be extracted from eye-tracking data. This information can potentially be leveraged to improve future interfaces and evaluate operator states in real-time. Although the pilot study is not extensive enough to produce statistically significant results, the methodologies presented provide valuable intuition and demonstrate the potential uses of eye-tracking within supervisory applications.

Robust Task-Scheduling Strategies for Multi-Operator Missions: Many supervisory missions require the operator(s) to sequentially process tasks (sensory data) that are generated by their autonomous partners. Proper scheduling of these tasks can have a profound impact on both operator and mission performance, since crucial mission planning decisions often rely on the operators' abilities to process tasks quickly and accurately [19, 20]. In the presence of multiple operators, it is necessary to both allocate

tasks and determine a processing order, evoking a combinatorial optimization problem whose solution is not straightforward. Traditional deterministic scheduling strategies are usually ill-suited for use in supervisory systems, due to (i) uncertainty in human behavior, and (ii) a failure to account for the operator's cognitive requirements, e.g., required levels of *arousal* or *stress*. In Chapter 3, we introduce a straightforward, flexible framework for operator task scheduling that accounts for processing time uncertainty and the imposed *task load* (related to *stress*) in order to optimize operator performance.

Part II: Sensor-Focused Methods

Cloud-Supported Coverage Control for Multi-Agent Surveillance Missions: Supervisory surveillance missions frequently require mobile agents (sensors) to periodically exchange data with a central *cloud* (in this context, the cloud coincides with the central data processing station/operator interface; see Figure 1.1). When operating in non-ideal environments or under hardware limitations, these potentially sporadic agent-cloud exchanges may be the only means of sharing real-time information across agents. In addition to missions involving UAVs, other applications that encounter this constraint include autonomous underwater vehicles that rely on periodic surfacing to communicate with a tower [21] and data mules that periodically visit ground robots [22].

In this type of cloud-based architecture, implementation of typical *decomposition-based* coverage control schemes (i.e., those that partition the workspace and assign each agent the coverage responsibilities of a single region) are not straightforward using existing approaches to dynamic workspace decomposition. Indeed, in cloud-based architectures, updated mission information is only relayed to one agent at a time, rendering traditional partitioning schemes, which rely on complete or pairwise coverage updates, impossible. Further, existing cloud-based strategies, e.g., [1], may introduce undesirable

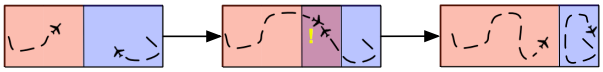

Figure 1.2: When complete or pairwise coverage updates are impossible, two updates are required to move from the left-most to the right-most configuration. When the red region is updated first, a collision (redundant sensing) risk is introduced.

configurations or collision-risks (Fig. 1.2). In Chapter 4, we seek to alleviate these issues by introducing a cloud-supported, decomposition-based framework for multi-agent persistent surveillance that promotes effective coverage without introducing collision (redundant sensing) risks and without requiring ideal or pre-planned data exchanges.

UAV Surveillance Under Visibility and Dwell-Time Constraints: Many supervisory applications, e.g., military operations [23], utilize a fixed-wing UAV to collect visual data within a large environment. In Chapter 5, we study a particular persistent surveillance mission in which a UAV that is equipped with a gimbaled camera is tasked with providing surveillance imagery of multiple static targets, each of which is associated with a pre-defined set of viewing constraints. The imaging constraints associated with each target include (i) a desired tilt angle with tolerances, (ii) a desired azimuth with tolerances, including the option of a 360-degree view, and (iii) the amount of time that the UAV should dwell before moving to the next target. The goal is to construct flight paths that are optimal in some sense while simultaneously allowing all imaging constraints to be satisfied. In this type of surveillance mission, there are often also multiple objectives, leading to a difficult, combinatorial optimization. We present a heuristic framework that systematically constructs UAV routes to image all targets to specification, and approximates solutions to the full, multi-objective routing problem.

Part III: Joint Methods

Joint Human/UAV Scheduling in Supervisory Surveillance Missions: As already noted, the presence of both human and autonomous elements within a single system is potentially very beneficial, since the interplay between these components can, in theory, create a symbiotic relationship that emphasizes the strengths and mitigates the deficiencies of each. However, this type of interaction is not guaranteed, even if each component is optimized independently. To realize the full benefit of this setup, coordination schemes must jointly optimize the *entire* system, accounting for the inherent coupling between the human and autonomous elements.

To demonstrate this approach, Chapter 6 considers a discrete surveillance mission (inspired by the mission in Chapter 5), in which a human operator analyzes imagery that is collected and transmitted in real-time by UAVs as they visit a set of discrete, geographically spaced targets. In contrast to purely operator-focused or purely sensor-focused approaches that coordinate a single system component's behavior individually, we develop a joint optimization scheme to coordinate both the human and the autonomous agents simultaneously, consequently addressing inherent couplings between them explicitly. In particular, the generated target imagery can be viewed as a set of "tasks" requiring operator attention; as such, in addition to vehicle routing issues, a constrained scheduling problem, similar to that of Chapter 3, emerges. Since real-time imagery is only available to the operator while a UAV loiters at a target, both the operator and the UAV resources are simultaneously required to complete each task. Chapter 6 develops a framework to coordinate these resources by jointly optimizing over UAV routes and the operator task-processing schedule with the goal of (i) maintaining the operator's task load within a high-performance regime, and (ii) minimizing unnecessary UAV loiter time.

1.2 Literature Review

Research relating to supervisory systems or their subcomponents takes various forms, drawing insight from one or multiple scientific fields including engineering, psychology, operations research, and computer science. As such, the body of relevant literature is vast. This section briefly reviews existing research relating to those theoretical topics that are most relevant to the particular coordination approaches studied herein.

Human Supervisory Control and Decision Supports: Many current applications utilize human-centered automation systems, such as dynamic positioning systems for maritime applications [24], command and control systems for monitoring space assets [25], automated vehicle operation aids [26], aviation accident and emergency response systems [27], numerous military technologies [4, 28], medical imaging systems [29], advanced traffic management and intelligent transportation systems [30], and many more.

As a consequence of the growing interest in human supervisory control, a large body of research has focused on the direct incorporation of human performance models into autonomous system design, and significant efforts have focused on finding systematic ways of distributing operator cognitive resources effectively. In some approaches, the human decision-making process is unregulated, but the automated system is tailored to the human operator's cognitive requirements. Research efforts focusing this approach often involve (i) optimal scheduling of operator tasks [31, 32, 33, 34, 35, 36, 18]; (ii) shortening of human reaction times through mediation of operator utilization [37, 38]; and (iii) efficient work-shift design to counter fatigue or interruption effects [39]. In other approaches, both the operator's decision-making process and the autonomous agents are controlled. For example, the human operator is given a set time to spend on each task, and operator decisions are used to alter automation schemes. Typical research efforts

under this approach focus on (i) determining optimal operator attention allocation both within and across tasks [40, 41, 42]; (ii) managing operator workload and/or task load [43]; and (iii) controlling autonomous agents to collect the most useful information [44, 45, 43].

With the ever-increasing maturity of sensing technology, many researchers studying human-centered systems study real-time, adaptive schemes, in which both physiological and performance measures are used to infer the operator's cognitive state, and automated functionalities are only triggered upon detection of non-optimal or undesirable states [46, 47]. However, the majority of such adaptive systems to date have been experimental rather than practical due to difficulties in accurately characterizing the user cognitive states [48]. Regardless, continually improving quality and affordability of physiological sensors, such as eye trackers and electroencephalography (EEG) devices, have led to improved metrics for objective analysis of real-time cognitive behavior [49].

Eye-tracking and Human Performance: Chapter 2 explores the use of eye-tracking to provide insight into operator behavior. Eye-tracking has been studied extensively in a variety of contexts, particularly cognitive psychology. Some eye-tracking literature studies the dynamics of eye-movements, e.g. [50]; although it has been shown that eye-movements are highly dependent upon the task that is being performed [51]. As such, works such as [52] have sought to incorporate top-down, i.e., task-dependent factors into eye-movement models. For our purposes, the most relevant works regarding eye-movements are those that study visual search [53], but even these results are sensitive to the specific search task being performed. Eye-tracking has also been used to aid in quantifying the mental state of the subject in question [54, 55], and as a feedback mechanism for the design of engineering systems [56, 57].

Supervisory System Design and RESCHU: Many research efforts have focused on

studying the effects of system design factors such as level of automation [58], the presence or absence of decision support systems [59], and the choice of control architecture [60] on human performance in supervisory tasks. In particular, RESCHU (see Section 1.1) has become a popular platform for testing and analysis of coordination schemes in the context of human supervisory control [18, 61, 62, 63]. The study in Chapter 2 relating to usability analysis of RESCHU is primarily motivated by [60], in which researchers use RESCHU to show the benefits on operator workload and performance of using task-based, as opposed vehicle-based, control. Here, *task-based control* refers to a setup where the user does not have direct control over individual vehicle trajectories and can only issue high-level commands, whereas *vehicle-based control* refers to a setup where the user directly controls each vehicle trajectory. Chapter 2 extends this work by using eye-tracking to better characterize user interactions with task-based RESCHU and further develop an intuition and understanding of operator interaction with a typical supervisory interface.

Discrete Task Scheduling in Human-Centered Systems: Typical discrete task-scheduling problems are NP-hard in general [64]. Despite its difficulty, the traditional deterministic scheduling problem and its variations have been considered for a number of years in the form of job-shop problems, e.g., [65], and a variety of high-quality heuristic methods exist for constructing effective solutions. Common solution strategies utilize integer programming [66], disjunctive graphs [67], and various heuristics [68]. Although these strategies are well-established for deterministic settings, they are often ill-suited for human-centered systems, primarily due to uncertainty in human behavior. In particular, processing times in human visual search usually carry significant uncertainty. Although some existing strategies consider resource allocation and discrete scheduling in uncertain or dynamic environments (e.g., [69]), and robust optimization methods are feasible in some circumstances [70], other factors that are generally unique to human-centered

systems can cause even these strategies to fail. For example, cognitive workload, fatigue, memory retention, among others, all have some effect on operator performance in persistent task execution missions.

The subproblems in Chapters 3 and 6 both contain a scheduling component that accounts for the operator's *task load*, which is rigorously defined as "a measurement of human performance that broadly refers to the levels of difficulty an individual encounters when executing a task" [71]. We focus on task load, as opposed to other human factors issues, since it has well-established trends and links to performance that can feasibly be exploited by mission planners [72]. In particular, the operator's task load is closely related to the more abstract notion of *stress* [73, 74], and, under the typical interpretation of the Yerkes-Dodson law [75, 76], moderate amounts of operator stress result in optimal operator performance. This logic motivates our strategy of maintaining the operator's task load within a moderate regime as a means of improving primary task performance.

Sensor-Focused Coordination Strategies for Multi-Agent Systems: A large amount of recent research focuses on the development of coordination strategies specifically for UAV applications (e.g., [77, 78, 79]). However, UAV research is a sub-class of a much larger body of literature that addresses algorithmic design and high-level reasoning for general autonomous mobile sensor applications [80]. These applications often necessitate solutions to complex routing problems, which may involve logical, temporal, and spatial constraints, as well as environmental uncertainty. When construction of global optima is not feasible, heuristics may still permit the real-time construction of solutions for use within practical systems. For example, variations on the classic Traveling Salesperson Problem (TSP) [81, 82, 83] arise frequently and, since the TSP is NP-hard, global optima usually cannot be consistently found in reasonable time. However, several new insights have been developed over the last decade for the classical version of the TSP,

along with a number of variations [84, 85, 86], and sophisticated heuristic solvers, e.g. [87] can quickly construct high-quality solutions for practical use. Other problems that commonly arise in mobile sensor applications include the construction of region selection policies [44, 88], the derivation of static and dynamic coverage schemes [89, 90], the development of persistent task execution schemes [91], and the design of load balancing strategies [92, 93].

Multi-Agent Coverage Control and Persistent Surveillance: Typical coordination strategies for multi-agent coverage control involve optimization [94], auctions [95], meta-heuristics [96], potential fields [97], Markov decision processes [98], among others [99]. The coverage control problem of Chapter 4 is related to *persistent surveillance* (*monitoring*), in which a sensor team is tasked with continual surveillance of some region, requiring subregions to be visited multiple (or infinitely many) times to minimize a cost, e.g., the time between visits or the likelihood of detecting events [100]. Persistent surveillance is a generalization of *patrolling*, where agents follow closed tours to protect or supervise an environment. Most solutions to patrolling problems utilize operations research, non-learning multi-agent systems, and multi-agent learning [101]; however, existing formulations are often one-dimensional and solutions usually reduce to "back and forth" motions that do not easily generalize, e.g. [102].

Decomposition-Based Surveillance: The framework proposed in Chapter 4 employs *workspace decomposition* to reduce a multi-agent coverage problem into a set of single-agent problems. This approach is common in multi-agent systems due to simplicity and scalability [100]. For planar persistent surveillance, decomposition-based approaches consist of two primary components: partitioning and single-agent routing. The most common approaches to optimal partitioning in convex environments are based on Voronoi

partitions [103], and effective schemes exist for constructing optimal partitions under various constraints [104, 105, 106]. Non-convex workspaces are typically addressed by creating a graph approximation, on which a number of graph partitioning schemes can be used [107]. In robotics, discrete partitioning is often considered under communication constraints [108, 1]. The scheme in Chapter 4 most closely mirrors [1]; however, in contrast to [1], our approach employs additional logic to ensure the resultant coverage regions retain properties that are consistent with effective, decomposition-based surveillance.

Single-agent path planners for persistent surveillance typically operate on graphs [109, 110, 44], and classical problems, e.g., TSPs [81], often play a key role in this case. Schemes for non-discrete spaces (open subsets of Euclidean space), are less common. Here, strategies include *a priori* construction of motion routines [111], adaptation of static coverage strategies [112], the use of random fields [113], and spectral decomposition [114]. The modular framework discussed in Chapter 4 incorporates any single-agent planner satisfying mild assumptions (Section 4.4).

Remarkably few papers explicitly address the implication of combining dynamic partitioning with continuous routing for multi-agent persistent surveillance. Existing work is mostly preliminary, considering ideal conditions and simplistic methods; e.g., the authors of [115] use a sweeping algorithm for partitioning and guide vehicles via lawn-mower patterns, while [116] uses rectangular partitions and a reactive routing policy. The authors of [117] use slightly more sophisticated partitioning along with lawn-mower trajectories. In [118], partitions are based on the probability of target presence, but ideal communication is assumed. Others, e.g. [119], use a decomposition-based approach, but focus on assignment without detailed treatment of the combined assignment/routing protocol.

Cloud-Supported Robotic Architectures: Chapter 4 considers a *cloud-supported* computational framework. Cloud-based robotic infrastructures (*cloud robotics*) have seen

growing research interest of late, as they can provide many benefits to complex systems, such as the storage and analysis of "big data," the availability of parallel grid computing, the potential for collective learning, and the utilization of human computation [120]. In multi-agent systems, cloud-supported schemes have been used for tasks such as collective optimization [121], rendezvous [122], and coordinated task-assignment [1]. In human supervisory control applications involving mobile sensors, cloud-supported architectures arise naturally, since mobile agents are typically required to transmit sensor data to a remotely located human operator for analysis (thus requiring a central repository), and harsh operational environments often prohibit reliance on peer-to-peer communication.

Discrete Surveillance and TSP Variations: Chapter 5 studies a discrete surveillance problem that is loosely interpreted as a generalization of both the *Polygon-Visiting Dubins TSP* (PVDTSP) [123, 124] and the *Dubins TSP with Neighborhoods* (DTSPN) [125]. The PVDTSP and the DTSPN are variations on the standard *Dubins TSP* (DTSP), requiring vehicles visit planar regions rather than discrete points. To incorporate imaging constraints, Chapter 5 adopts a strategy that is, in some sense, an extension of [124], where the authors approximate solutions to a PVDTSP by discretizing the polygons and posing a *Generalized TSP* (GTSP) (also called the *Set TSP*, *Group TSP*, *(Finite) One-in-a-Set TSP*, *Multiple Choice TSP*, or *Covering Salesperson Problem*). Discretization-based strategies that approximate a continuous motion planning problem with a discrete path-finding problem over a graph are typically called *sampling-based roadmap methods* [126, Ch. 5]. Such methods are traditionally used for point-to-point planning among obstacles; however, they have also been used for more general Dubins path planning, e.g. [127]. Like the TSP, the GTSP is a combinatorial optimization; however, strategies exist for computing high-quality solutions. The most popular approach converts the GTSP into an *Asymmetric TSP* (ATSP) using a *Noon-Bean transform* [128]. The availability of

efficient ATSP solvers, e.g., LKH [87], make such transformations a practical GTSP solution option even though they do not produce global optima in general. Other common GTSP solution approaches make use of meta-heuristics, e.g. [129].

Multi-Objective Optimization: The scheduling problem of Chapter 3, the discrete surveillance problem of Chapter 5, and the joint scheduling/routing problem of Chapter 6 all consider optimization problems that, in some sense, contain multiple, potentially conflicting objectives. Multi-objective optimization problems are well-studied in existing literature [130, 131, 132]. In engineering, most solutions attempt to satisfy some predetermined notion of optimality, the most common of which being *Pareto optimality*. Pareto-optimal fronts are usually difficult to characterize directly, so they are typically constructed through *scalarization*, which maps the solutions of a related single-objective optimization problem to Pareto-optimal solutions of the multi-objective problem. The most common scalarization techniques are linear scalarization, where the cost is a linear combination of the objectives, and the ϵ-*constraint method*, where the values of all but a single objective are explicitly treated as optimization constraints [133]. We note that no single approach to multi-objective optimization is superior in a general sense; rather, the appropriate method depends on the type of information available, the user's preferences, solution requirements, and the availability of software [131].

Joint Human/Sensor Optimization for Supervisory Control: Chapter 6 presents a scheme that jointly optimizes both the UAV and operator behavior. Despite extensive research devoted to improving each component individually, there have been very few attempts to jointly optimize both operator and autonomous agent behavior. Indeed, existing work typically assumes a "loose" coupling between the human and autonomous agents, resulting in *reactive* policies. For example, the authors of [38, 40] assume that op-

erator tasks arrive in a queue according to a fixed stochastic process, which determines the optimal processing times. The authors of [44] use an adaptive surveillance policy based on operator responses in a target detection task; however, operator behavior is not controlled. Some work, such as [18], develops discrete-event simulation models, which can be used for testing and optimization. However, these abstractions utilize parametric process models that do not explicitly consider the cause of task generation.

Very limited work has considered more tightly coupled coordination. Those most closely related to the study in Chapter 6 are [134, 135], which use optimization schemes for simultaneous routing and scheduling under operator workload constraints. However, these works contain several limitations. In particular, both [134] and [135] rely on the availability of a suitable abstraction to the vehicle routing problem, which may be unavailable in complex scenarios. For example, [134] requires an accurate predictor of the mission cost associated with a given task-agent pairing, which is often unavailable *a priori* in non-trivial problems. In addition, both approaches consider deterministic setups; as such, solution quality can become poor or even infeasible during mission execution when operator uncertainty is introduced. With respect to [135], all planning operations are performed offline, and may not be easily applied to dynamic schemes due to computational complexity. Indeed, the authors of [135] discretize the time continuum to obtain a pure integer program approximation, which can quickly become intractable for even modest time horizons. In contrast, the distributed framework in [134] is designed for online implementation; however, it does not consider tasks requiring both human and robotic resources simultaneously. Chapter 6 presents a joint optimization framework that seeks to overcome the above limitations by: (i) combining vehicle routing and operator scheduling into a single, mixed-integer problem; (ii) providing a scalable online/incremental implementation methodology, and (iii) demonstrating a natural extension that incorporates robustness to uncertainty in operator processing times.

1.3 Contributions

The primary contribution of this dissertation is the illustration of a number of fundamentally different approaches to generating effective coordination schemes for use in human supervisory control systems involving autonomous mobile sensors. Although the discussion herein focuses on particular subproblems (primarily motivated by applications in which the operator(s) analyze visual data generated by UAVs), the presented content also serves to demonstrate various design philosophies, which each draw from numerous scientific fields, that can be applied to a broader set of human-centered systems. Indeed, Part I focuses on improving operator performance, primarily using tools from psychology, human factors, computer science, and operations research; Part II focuses on improving mobile sensor performance, primarily using tools from engineering, control systems, and robotics; and Part III focuses on joint methods that combine operator-focused and sensor-focused methods within a single optimization framework.

More specifically, individual chapter contributions are summarized as follows.

Chapter 2 illustrates how eye-tracking metrics can be used to both analyze operator behavior and assess the usability of supervisory control interfaces. The main contribution of this chapter is the inclusion and analysis of eye-tracking data from a pilot usability study of the task-based RESCHU interface. For the study, we collected gaze and pupil size information using a non-restrictive eye-tracker while human subjects performed a search and surveillance mission involving either 4 or 8 UAVs. We provide analysis of 2 complete eye-tracking data sets, one from a user on the 4 UAV condition and one from a user on the 8 UAV condition. We find both qualitative and quantitative differences in gaze characteristics (i) between the two users, and (ii) between events occurring when the user was engaging in a target search, and those occurring at other times in the experiment.

Specifically, the main results of this chapter are as follows: First, by dividing the user interface into disjoint regions of interest, we calculate statistics including the average time that the user's gaze fell upon each region, the distribution of fixation lengths in each region, and the fixation transition probabilities between regions under the 1-step Markov assumption. Second, by comparing these statistics across the the search and non-search portions of the experiment, we are able to postulate as to what a gaze pattern would be for a "typical user" and suggest possible interface improvements. Third, we analyze pupil statistics, finding that pupil size was smaller during visual search tasks than during nonsearch tasks. Finally, we discuss numerous valuable future research directions.

Chapter 3 addresses the problem of determining the optimal processing schedules for general, multi-operator sequential task analysis. Specifically, we build a MILP framework for a multiple operator task-scheduling problem that incorporates task load considerations. We then illustrate how robustness to task processing time uncertainty can be added into this formulation through the use of scenarios. We also demonstrate the flexibility allowed by this scenario in choosing both the desired degree of robustness and the degree that task load is considered. Next, we illustrate how, in certain situations, general performance can be improved through the use of adaptive schemes, which employ both strategic re-planning and estimation. We develop a receding horizon re-planning strategy for single operator scheduling, and subsequently expand our problem formulation to include an additional estimation step. We then discuss adaptations for use with multiple operator setups. Finally, we show through simulation how increased computation times due to such extensions may, in some scenarios, make some of the presented schemes intractable for practical use in their raw form. As an alternative, we propose a heuristic for task assignment and show that we can achieve vast computational advantages at minimal performance expense. Throughout our discussion, we also address issues that arise

in practical implementation of our proposed framework.

In **Chapter 4**, we shift our focus to the development of coverage control schemes for multi-agent systems that operate in dynamic and communication-constrained environments. We develop a cloud-supported, decomposition-based, multi-agent coverage control framework for persistent surveillance, which requires only sporadic, unscheduled data exchanges between agents and a central cloud. In particular, we develop a sophisticated partitioning and coordination scheme that can be effectively paired with single-agent trajectory planners. This leads to the complete, modular framework in which high-level coverage is coordinated on the cloud and agent trajectories are generated independently via on-board planners. We encompass realistic constraints including restrictive communication, dynamic environments, and non-parametric event likelihoods.

Specifically, our dynamic partitioning scheme only requires agents to sporadically upload and download data from the cloud. The cloud runs updates to govern region assignments, while also manipulating surveillance parameters. We prove that these updates produce desirable properties: coverage regions collectively form a connected N-covering and evolve at a time-scale that allows for appropriate agent reaction, no subregion remains uncovered indefinitely, local likelihood functions have disjoint support, among others. In certain cases, we show that the set of coverage regions and generators converges to a Pareto optimal pair in finite time. We show that the combination of our partitioning scheme with a trajectory planner ensures collision avoidance, provided the planner obeys natural restrictions. We illustrate our framework through numerical examples.

We note that this partitioning approach is primarily motivated by [1]; however, the algorithms proposed in Chapter 4 are explicitly designed to operate within a multi-agent surveillance framework and introduce additional logic that evokes a set of desirable properties. The proposed scheme has the following advantages: First, our framework main-

tains connectivity of intermediate coverage regions, ensuring that agents can visit their entire assigned region without entering another agent's territory. Second, our framework provides inherent collision avoidance when the scheme is paired with low-level motion planners. Third, our algorithms explicitly manipulate local likelihood functions maintained by the agents to guarantee that each has support within an allowable region, promoting seamless and modular pairing with any trajectory planner that uses the support of the event likelihood to govern agent routes, e.g., [114]. The framework has these features while maintaining similar convergence properties as the algorithms in [1].

Chapter 5 addresses a different persistent surveillance mission in which fixed-wing UAVs are required to visit a set of discrete targets (rather than planar regions as in Chapter 4). We show how the multi-objective routing problem with both visibility and dwell-time constraints can be rigorously posed as a constrained optimization problem under reasonable assumptions on UAV behavior. Indeed, we define *visibility regions* at each target, which reflect imaging requirements, along with a set of feasible dwell-time maneuvers to be performed within the regions. These constructions are then translated into a set of constraints that are incorporated into a precise problem statement, which represents an ϵ-constraint scalarization of the multi-objective problem. We then approximate the constrained optimization problem, having an infinite solution space, with a discrete problem, having a finite solution space. The discrete approximation implicitly considers both the required dwell-times and the visibility constraints, as it is formulated through a selective discretization procedure. Next, we present a novel heuristic method for solving the discrete approximation that leverages solutions to GTSP instances. We show that this method produces feasible solutions and, in many cases, maps optimal solutions of a related GTSP directly to optimal solutions of the discrete problem. Finally, we incorporate these constructions into a complete heuristic framework to produce high-quality

solutions to the full, multi-objective routing problem. We prove that the heuristic algorithm is *resolution complete* in a specific mathematical sense. Then, we present a greedy heuristic method for expanding the heuristic for use in multi-vehicle problems. We finish by testing these methods numerically.

We note that the work in Chapter 5 is most closely related to that presented in [124], which uses a similar sampling-based framework to address a PVDTSP. However, the framework in [124] considers a single metric (total tour time), and cannot incorporate non-trivial dwell-times. As such, our work is can be viewed as an expansion of [124] to incorporate a more general set of imaging behaviors, and accommodate an additional performance metric that is reflective of realistic mission scenarios.

Finally, **Chapter 6** shows how a tighter coupling between the human and autonomous components can be achieved through joint optimization. In particularly, we develop a joint optimization framework for a discrete, supervisory surveillance mission similar to that of Chapter 5, in which UAVs must visit a set of discrete targets and transmit visual data to a remotely-located operator. The chapter opens by formulating the multi-vehicle, scheduling/routing problem as a Mixed-Integer Non-linear Program (MINLP), whose objective function captures both the task load imposed on the operator and the time that UAVs spend loitering unnecessarily. We show that the general MINLP can be re-formulated as a MILP, at the expense of a significant increase in the problem size. In single vehicle missions, we show that an alternative linearization exists that does not affect the problem size. Next, to ease computational issues, we introduce a receding-horizon framework for constructing suboptimal solutions. Here, whenever the operator finishes processing a task, a re-planning operation is initiated that only chooses each UAV's next destination and the impending portion of the operator's schedule. We pose this finite horizon re-planning operation as a comparatively small-scale MILP, whose solutions are

constructed using existing solvers. We then show how this receding horizon framework can be manipulated to provide robustness to uncertain processing times via a straightforward, scenario-based extension of the re-planning MILP. Finally, we demonstrate the utility and flexibility of our framework in a set of simulated missions. In addition to illustrating the performance and robustness properties of the proposed solution, this example also demonstrates how the dynamic solution framework readily extends to incorporate more general problem setups such as those containing fixed-wing UAVs.

1.4 Permissions and Attributions

1. Portions of Chapter 1 and the complete content of Chapters 4, 5, and 6 are the result of a project sponsored by the Institute for Collaborative Biotechnologies at the University of California, Santa Barbara (UCSB) and carry the following disclaimer:

   "This work has been sponsored by the U.S. Army Research Office and the Regents of the University of California, through Contract Number W911NF-09-D-0001 for the Institute for Collaborative Biotechnologies, and that the content of the information does not necessarily reflect the position or the policy of the Government or the Regents of the University of California, and no official endorsement should be inferred."

2. Portions of Chapter 1 are the result of a collaboration with Vaibhav Srivastava, Miguel Eckstein, Amit Surana, Grant Taylor, and Francesco Bullo, and has previously appeared in IEEE Control Systems Magazine [136]. This content carries the following notice: ©2015 IEEE. Reprinted, with permission, from Jeffrey R. Peters, Vaibhav Srivastava, Miguel P. Eckstein, Amit Surana, Grant S. Taylor, and Francesco Bullo, *Human Supervisory Control of Robotic Teams: Integrating Cogni-*

tive Modeling with Engineering Design, November 2015. `http://www.ieee.org`.

3. The content of Chapter 2 and portions of Chapter 1 are the result of a collaboration with Luca Bertuccelli and Amit Surana. The original (previously unpublished) report from this work is available at `https://arxiv.org/abs/1506.01976`.

4. The content of Chapter 3 and portions of Chapter 1 are the result of a collaboration with Luca Bertuccelli, and have previously appeared in the AIAA Journal of Aerospace Information Systems [137]; reproduced with the permission of the American Institute of Aeronautics and Astronautics, Inc.: `http://www.aiaa.org`.

5. The content of Chapter 4 and portions of Chapter 1 are the result of a collaboration with Sean Wang and Francesco Bullo, and have previously appeared in ASME Journal of Measurement, Control, and Dynamics [138]; reproduced with the permission of the American Society of Mechanical Engineers (ASME): `http://www.asme.org`.

6. The content of Chapter 5 and portions of Chapter 1 are the result of a collaboration with Amit Surana, Grant Taylor, Terry Turpin, and Francesco Bullo.

7. The content of Chapter 6 and portions of Chapter 1 are the result of a collaboration with Amit Surana and Francesco Bullo.

Part I

Operator-Focused Methods

Chapter 2

Human Modeling and Usability Analysis Using Physiological Sensing

This chapter explores the use of physiological sensing, specifically eye-tracking, for the purpose of better understanding operator interactions with supervisory control interfaces. In particular, we study the task-based version of the Research Environment for Supervisory Control of Heterogenous Unmanned Vehicles (RESCHU).

2.1 The RESCHU Interface

We first give a brief overview of task-based RESCHU. For a comprehensive description, we instruct the reader to consult [18, 60]. The RESCHU environment allows a single user the ability to control multiple UAVs in a search and identification task. Figure 2.1 shows a screen shot of the Graphical User Interface (GUI), which has 6 main features: (1) a map showing positions of UAVs, targets, and hazard areas, (2) a search window which displays the UAV payloads (camera imagery), (3) a message window which relays system information to the user, (4) a panel for engaging the payload, i.e., displaying the

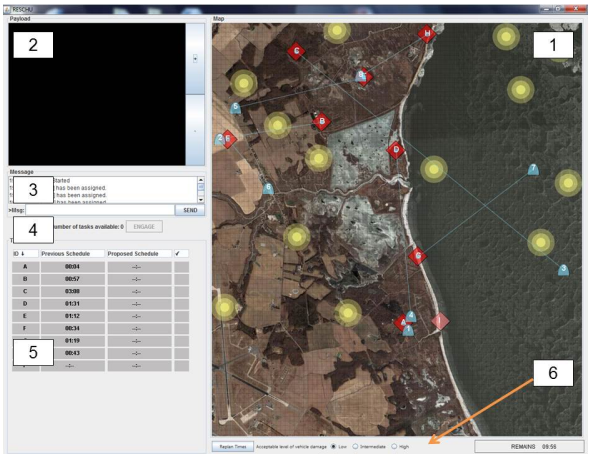

Figure 2.1: The RESCHU interface and its main features: (1) the map window, (2) the search window, (3) the message window, (4) the engage panel, (5) the scheduling panel, and (6) the re-plan panel.

camera imagery to the user, (5) a table showing estimated arrival times for the UAVs to their assigned tasks, and (6) a panel for selecting damage tolerances and re-planning UAV trajectories. The map depicts the UAVs as bullet shapes according to MIL-STD-2525C convention, and depicts hazard areas and task locations as yellow circles and diamonds, resp. The UAVs incur damage when they intersect a hazard area. The hazard areas stochastically appear and disappear, which creates a need for dynamic path planning.

Once a UAV reaches a task location, an "ENGAGE" button on the panel (4) becomes active. When the operator presses the button, a surveillance image appears in the search window (2). Images are static, but can be panned/zoomed (see Figure 2.2). A textual description of the search target appears in the message window (3). Once the operator believes that they have found the target, they right-click where they believe the target object to be. Upon task completion, the UAV is automatically re-assigned to a new task and immediately begins moving in a straight-line path to its new destination.

The operator also has the option of changing the UAV flight paths to avoid damage. We focus on a task-based control setup [60]. The operator is shown current task assignments for all UAVs in the map window (region (1), Figure 2.1). The operator can only

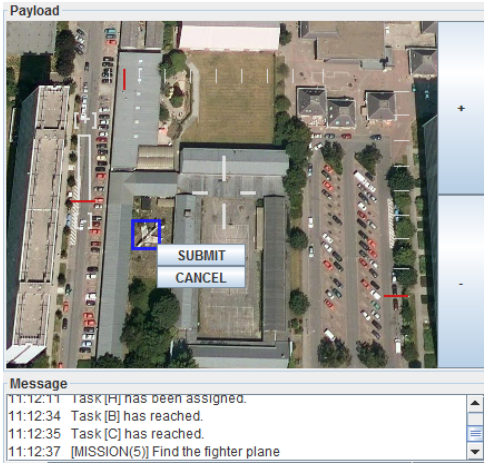

Figure 2.2: Example UAV imagery. The target is specified in the message panel, and the user is able to pan and zoom in the available image by clicking on the image and using the radio buttons, respectively. Once the user finds the target, they can right click on the target and select "SUBMIT".

change assignments and/or flight paths by selecting a damage tolerance (low, medium, or high) in the replan panel (region (6), Figure 2.1) and clicking the "Replan Times" button. The system then calculates estimated arrival times for all vehicles. If the operator likes the new plan, they accept by clicking the check mark in the scheduling panel (region (5), Figure 2.1), and new arrival times are automatically tabulated.

2.2 Experimental Setup

2.2.1 Participants

We collected eye-tracking data on 4 subjects as they interacted with the simulation (all male, age in range 18-44). However, due to experimental error, we only were able to obtain complete, reliable eye-tracking data sets from 2 out of the 4 users, one of which performed a search task involving 4 UAVs (Participant 3, age in range 18-34, no prior experience with RESCHU) and the other performed a task involving 8 UAVs (Participant 4, age in range 18-34, extensive experience with RESCHU, i.e. "expert user"). Since the

emphasis of this chapter is the analysis of eye-tracking data, we focus on these two users. Both of these subjects indicated that their occupation was related to science and engineering. Although 2 eye-tracking data sets is a very small sample, the objective of this pilot study is not to provide conclusive statistical insight, but rather to provide test data and intuition about supervisory interfaces that can guide researchers in future experiments.

2.2.2 Methods

The hardware setup consisted of a computer equipped with a Tobii x120 eye-tracking mechanism [139]. The subject was first instructed to complete a training module on how to use the RESCHU interface. Once the user felt ready, the experimental investigator guided the subject through calibration of the eye-tracker and started the simulation.

Each trial lasted 10 minutes, during which the participant interacted with RESCHU in the manner described in Section 2.1. Participants had 2 main goals: (i) correctly process as many tasks as possible, i.e., find as many targets as possible, and (ii) incur the least amount of UAV damage by the end of the mission. Time stamps for virtually all events (engagement of search tasks, etc.) during the simulation were logged by the RESCHU software. The eye-tracking mechanism did not make physical contact with the participants and did not impede their ability to interact with the interface. The eye-tracker collected data at a rate of 60 Hz. Data that was logged by the eye tracker included a time stamp, horizontal and vertical positions of the subject's gaze on the screen (px), and pupil sizes (mm). In addition, the eye tracker automatically classified each gaze event as being either a saccade, a fixation, or an unclassified event using a built-in filter (see [139]). With this information, the eye-tracker sequentially numbered each gaze event and assigned a corresponding duration. Finally, a validation vector was

Table 2.1: Summary of Usability Study Results

Metric	Findings
Gaze Events	• Qualitative and quantitative differences between search/nonsearch • "Unbalanced" in search times, "balanced" in nonsearch • Qualitative and quantitative differences between users • 8 UAV user (expert) had more gaze transitions during search times • Event durations follow right-skewed distributions
Pupil Size	• Lower during search times for both users

included which assigned an integer between 0 (low uncertainty) and 4 (high uncertainty) according to the quality of each measurement.

2.3 Results

In what follows, we refer to each occurrence of the user engaging a UAV payload, i.e., looking at camera imagery, and subsequently searching for a target as a *search task*. We quantify the length of one search task as the time between the user pressing the "ENGAGE" button and the user submitting the target location. We refer to times during which the user is engaged in a search task as *search times*. We say that the user is engaged in a *nonsearch task* if they are not performing a search task, and we refer to the corresponding times as *nonsearch times*. An brief overview of our main results is presented in Table 2.1 for convenience.

For participant 3, the eye-tracking mechanism logged $37,606$ data points, of which $29,503$ (78.3%) were logged during search times. For participant 4, the expert user, the eye-tracking mechanism logged $37,088$ data points, of which $26,087$ (70.3%) were logged during search times. Approximately 95% and 86% of the data points for participants 3 and 4, resp., were accurate according to the validity vectors provided by the eye-tracker.

During fixations, the eye-tracking hardware assigned each fixation a single "fixation

Table 2.2: Distribution of Gaze Events

Participant 3			
	Search	Nonsearch	Total
Map Window	4.2%	50.9%	16.4%
Search Window	89.7%	11.7%	69.4%
Message Window	1.9%	22.2%	7.2%
Engage Panel	0.0%	4.3%	1.2%
Scheduling Panel	0.1%	4.9%	1.4%
Replan Panel	0.0%	0.0%	0.0%
Unclassified	4.0%	5.9%	4.5%
Participant 4			
	Search	Nonsearch	Total
Map Window	1.0%	43.9%	13.7%
Search Window	78.3%	3.6%	56.2%
Message Window	9.1%	8.7%	9.0%
Engage Panel	3.2%	6.5%	4.3%
Scheduling Panel	1.3%	23.8%	8.0%
Replan Panel	0.4%	3.6%	1.3%
Unclassified	6.7%	9.8%	7.6%

point" by taking an average of the gaze locations at each data point during the fixation event. Because gaze location can vary slightly during fixations, we filtered data by replacing raw gaze locations during fixation events with the fixation point. Table 2.2 shows the percentage of gaze events during search, non-search, and total times, resp., falling within each region (Figure 2.1). Note that the similarity in the proportion of participant 4's gaze that fell in the search window during search times (78.3%), and proportion of participant 3's data corresponding to search times (78.3%) is coincidental.

For both participants, the percentage of gaze events falling within either the search window or the message window is much higher during search times (participant 3: 91.6%, participant 4: 87.4%) than during nonsearch times (participant 3: 33.9%, participant

4: 12.3%). During nonsearch times, both participants spent the most time looking in the map window (50.9%, 43.9%). If we examine the combinations of windows with the most gaze events during non-search tasks, participant 3 spent most of his time looking within either the map window, search window, or the message window (84.8%), while participant 4 distributed his gaze mostly among the map window, the scheduling panel, and unclassified regions (81.1%). Neither of the participants looked at the replan panel very often, with a negligible amount of participant 3's gaze events falling within this window, and only 1.3% of participant 4's total gaze events occurring in this region.

In order to analyze scan-paths, i.e., sequences of fixations, we consider the fixations that occurred in each of the 6 regions indicated in Figure 2.1. Assuming that the sequence of fixations satisfies the 1-step Markov assumption (valid in certain circumstances [15]), then we can construct a transition probability matrix by considering the probability that the next fixation will fall within a particular region of the screen, given the location of the current fixation. Figure 2.3 shows graphical representations of the fixation transition probability matrices for participant 3 and participant 4 for (i) the overall mission, (ii) search times, and (iii) nonsearch times. In these plots, the six regions are represented by blue boxes, each with an orange vertex placed randomly inside. The vertex located outside the boxes represents all unclassified regions. Lines connecting the vertices represent transitions between regions, with the direction of convex curvature representing a forward transition. The size of the vertex corresponds to the probability of a "self-loop", that is, that the next fixation will fall in the same region as the current fixation . For clarity, if the total number of transitions starting from a given region was less than 5, these transitions were omitted from the diagram. Notice the major qualitative differences between scan-path behavior of the two conditions (search or nonsearch). Both participants exhibited an increased probability of a self-loop in the search window, and a decreased probability of self-loops in the map window during search times. Further, notice that the transition

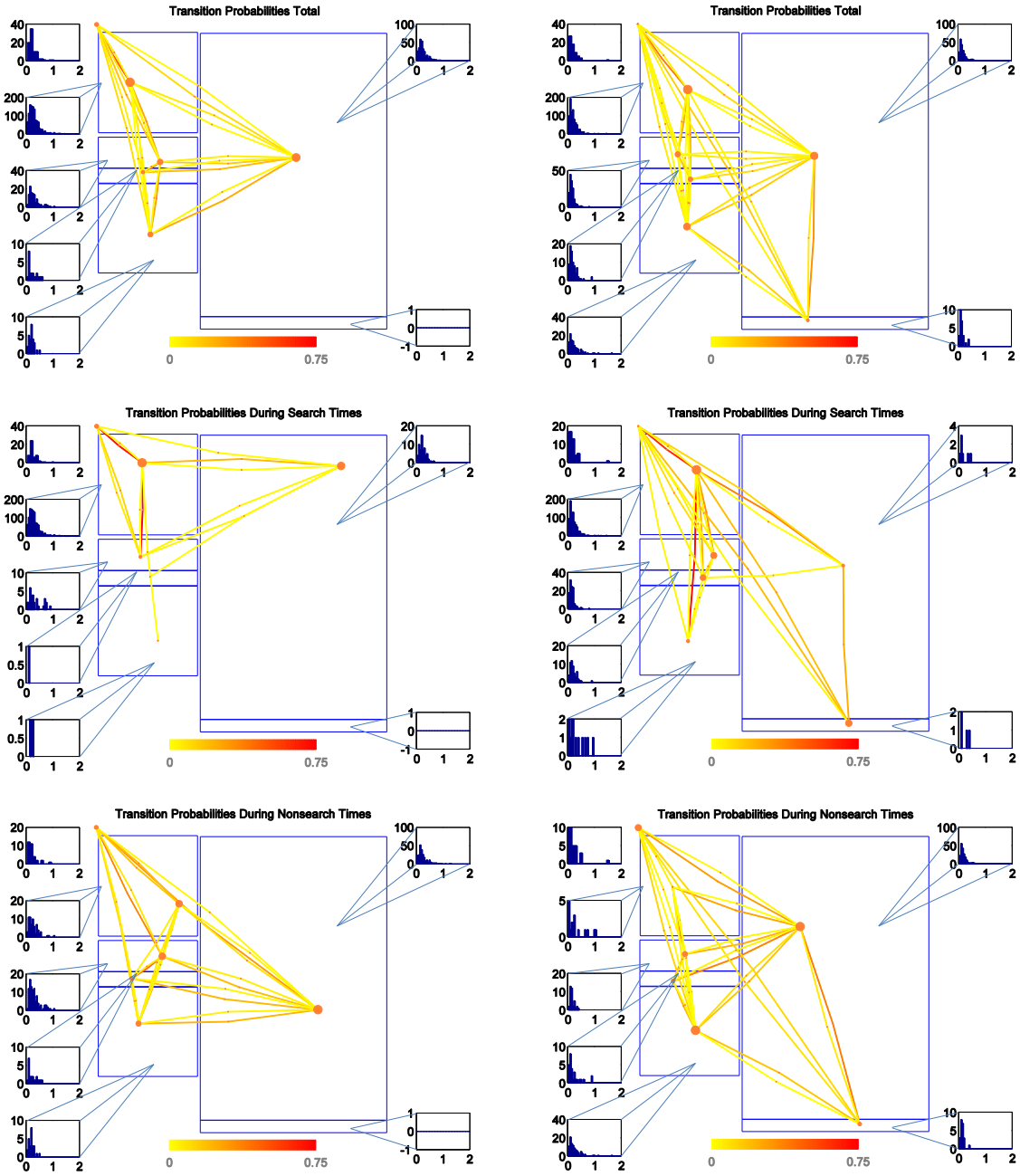

Figure 2.3: Fixation transition percentages for participant 3 (left column) and 4 (right column). The top two plots show the fixation transition probabilities for the experimental trial as a whole. The middle two plots show fixation transition probabilities for search times. The bottom two plots show the fixation transition probabilities for nonsearch times. Each plot additionally shows histograms representing the fixation lengths (s) that occurred in each region for each condition.

probability distribution is much more "balanced" during nonsearch times for both participants, i.e., there are much more equally distributed probabilities with respect to the transitions between regions. If we compare probabilities between the two participants, participant 4 showed a slightly higher tendency to transition to and from the scheduling window and unclassified regions than did participant 3, especially during search times. However, similar qualitative behaviors emerge across participants in all conditions. From a quantitative point of view, we consider the balanced Kullback-Leibler (KL) divergence as a metric between two Markov transition matrices A, B:

$$D_{\mathrm{KL}}(A||B) = \sum_{i,j} A_{i,j} \log\left(\frac{A_{i,j}}{B_{i,j}}\right) + B_{i,j} \log\left(\frac{B_{i,j}}{A_{i,j}}\right). \tag{2.1}$$

To produce finite values, we replace the 0 entries in each matrix with $\epsilon = 0.01$. Using this metric to compare across the participants, we obtain a KL divergence between the total transition matrices of 6.17, a KL divergence between the search transition matrices of 16.02, and a KL divergence between the nonsearch transition matrices of 9.80. Comparing within participants 3 and 4, resp., the KL divergences between the search and nonsearch matrices are 9.52 and 20.92, the KL divergences between the total and nonsearch matrices are 1.95 and 6.80, and KL divergences between the total and search matrices are 4.76 and 6.96. From these numbers, it is apparent that there are, in fact, quantitative differences between the search and nonsearch conditions. In addition, although the two users show qualitatively similar patterns in gaze transitions, they are still quantitatively different. In addition to the Markov transition matrices, Figure 2.3 contains histograms quantifying fixation lengths within each region. For clarity, all of the histograms were plotted on the same horizontal scale, with outliers omitted. Notice that all conditions produce distributions that are skewed to the right to varying degrees.

Plots of pupil diameter as a function of time are shown in Figure 2.4. In the figures,

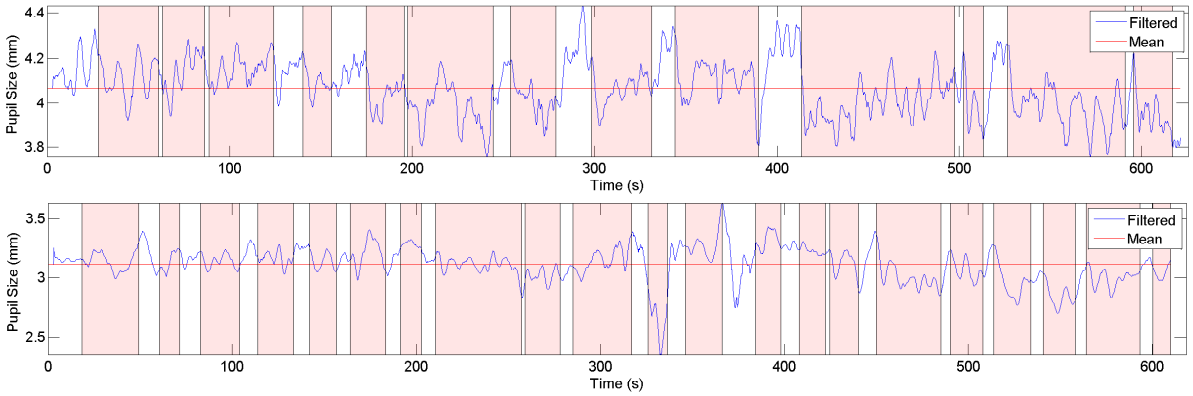

Figure 2.4: Filtered mean pupil sizes as a function of time (s) for Participants 3 (top) and 4 (bottom). Pink shaded regions in the plots correspond to search times.

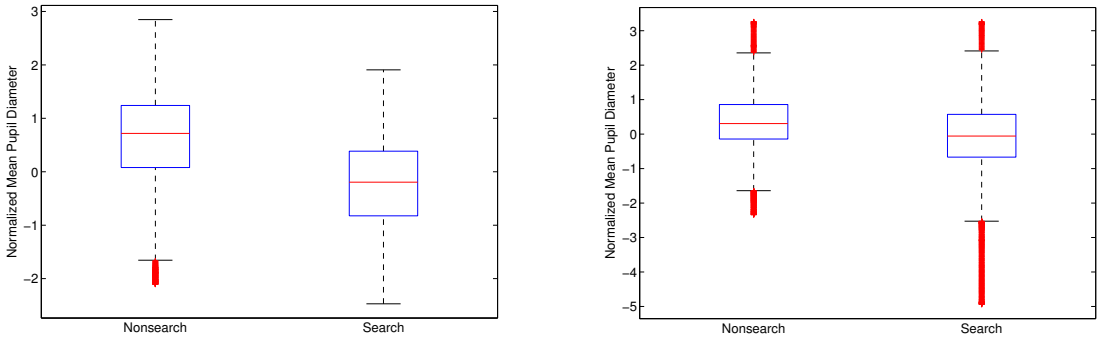

Figure 2.5: Boxplots showing the statistics for the mean normalized pupil diameters during search and nonsearch tasks for participant 3 (left) and 4 (right).

times during which the user was engaged in a search task are indicated by the pink shaded regions. The blue curve represents the filtered mean pupil diameter, i.e., the mean pupil diameter between the left and right eyes, while the red line represents the mean of the blue curve. Data was filtered by discarding non-valid points, and using a moving window filter with a window length of 180 points (3 s). Notice that for both users, pupils tend to shrink at the onset of search tasks. To further investigate differences in pupil size during search and nonsearch times, we normalize and re-scale the data points from each participant to have 0 mean and unit variance, and consider the pupil diameters that were measured in each condition. The result is shown in Figure 2.5. Assuming our data is

independent, and that the distributions are sufficiently normal due to the large numbers of data points, we can compare the means of the search and nonsearch pupil diameters via a 2-sample Welch's t-test (not assuming equal variances). This test reveals a significant difference between the search and non-search tasks for both participant 3 ($t = -78.93$, $df = 15699$, $p < 0.001$) and participant 4 ($t = -39.86$, $df = 24435$, $p < 0.001$).

2.4 Discussion

The fact that our hardware did not produce valid data sets for two of the users already presents valuable insight as to how to improve the RESCHU GUI. The two users that did not produce valid data sets tended to squint their eyes and lean in toward the screen, preventing the eye-tracker from collecting adequate data. The users both indicated that the reason for this behavior was that text on the interface was too small, and that they had difficulty seeing certain portions of the screen (particularly the search window). The eye-tracking data that was collected, particularly the data that is contained in Table 2.2 and Figure 2.3, also provides valuable insight as to how the interface should be changed. Specifically, since both of the users showed similar trends in the amount of time they spent looking in the various regions, it is conceivable that re-sizing the various windows to be more consistent with the proportions of the gaze events that occurred in the windows could improve performance. For example, the current GUI has the scheduling window taking up a large portion of the screen, but from Table 2.2 we see that user 3 focused on this window in only a very small part of the experiment, regardless of whether he was searching or not searching. User 4 also spent a relatively small amount of time looking at the scheduling window, especially during searches. Thus, it could be advantageous to make this window smaller. The differences that are present between the search and nonsearch conditions in the gaze-patterns in Fig. 2.3 also indicate that using

a dynamic re-sizing scheme to adjust the windows during these different tasks might improve performance. In this case, our data suggests that blowing up the search window during the visual searches and shrinking the other windows, especially the scheduling panel and the map window, could have beneficial effects.

Since both users qualitatively showed similar patterns in their gaze transition probabilities, Figure 2.3 allows us to postulate as to "typical" use of RESCHU. During non-search tasks, both users spent most of their time looking at the map window, but also spent a significant portion of the time shifting their gaze among different regions. During search times, the users fixated the most on the search window, with very few transitions out of this window. When the users did transition their gaze out of the search window during a search task, they generally transitioned back to the search window rather quickly. This suggests that typical use involves a balanced gaze approach in which the user constantly scans the screen (with slight bias toward the map window) during non-search times, and an unbalanced approach during search times, where the user spends almost all of their time in the search window with occasional short looks outside.

Despite qualitative similarities, our quantitative approach showed that there are still differences in overall gaze behaviors between the users which could potentially be exploited by system designers. The largest quantitative difference when comparing across users occurred between their respective search transition matrices. User 4 (the expert user) seemed to have a slightly more balanced approach in the sense previously discussed during search times, then did user 3. This could be a result of increased workload for user 4, since the increased tendency to transition outside the search window during search times could be a result of pressure to attend to other UAVs. Future research should investigate whether these differences in scan-paths during search tasks exist in other users, and if encouraging users in high workload conditions to focus on the search window does, in fact, improve performance. Differences in scan-paths across workload conditions could

also be useful in quantifying operator cognitive workload in real-time.

From a modeling perspective, the data contained in Figure 2.3 has the potential to produce a multi-layered user model. Indeed, at the highest level the user can be modeled as a state machine, with states such as "search" and "nonsearch", while within each state we can potentially model the user behavior on a finer level using relations similar to those contained in Figure 2.3. The utility of such a model should be assessed in future research. It is also of interest to expand the number of states in the finer level Markov chain construction to include the UAV locations in addition to static regions of interest.

The pupil diameter phenomena in Figure 2.5 provides intriguing insight. Most researchers in cognitive psychology generally have found that pupil diameter tends to increase during mentally challenging tasks. In our experiment, we observed *decreased* pupil diameters during the search tasks relative to those measured during the nonsearch tasks. One possible explanation for this is that visual search is mainly perceptual, as opposed to cognitive, and thus may not follow the same trends. Another possible explanation for our observed phenomena lies in the size of the windows and of the visual stimuli. It is known that pupil size is linked to visual acuity, i.e., how well a person can resolve stimuli of a given spatial frequency. In particular, some works, e.g. [140], have demonstrated inverse relationships between pupil size and spatial frequency. With respect to our experiments, all of the visual search stimuli (intentionally) contained high spatial frequencies and all of our participants, including participants 3 and 4, indicated that they felt as though the search window was too small and the map window was too big. Since our gaze data revealed that the user fixated mostly on the map window during non-search tasks and the search window during search tasks, it is possible that the observed pupil diameters are a reaction to changes in spatial frequency of the primary stimuli during search and nonsearch tasks. With this point of view, our data aligns well with previous research on pupil diameter phenomena. It is of interest to see if this pattern persists in other GUIs.

2.5 Chapter Summary

An understanding of operator behavior is often a key component to the development of effective supervisory systems. The analysis of this chapter suggests how eye-tracking data can potentially be used to deduce the user state and predict anomalous behavior, which, in turn, can be used to design decision supports. Indeed, from the eye-tracking data, we are able to postulate "typical use," which could be used as a basis of caparison for real-time systems. Further investigation is needed to determine if the postulated behavior is observed for general users and what changes emerge when the GUI is altered. Our data also suggests that scan-paths, particularly during search tasks, could be an indicator of cognitive workload. Theoretically, it is advantageous to further explore more explicit representations of scan-path behavior. It is also beneficial to characterize scan-paths within search images and investigate correlations with performance. The incorporation of fixation times on each sensor into scan-path models should also be investigated.

Our data also suggests that pupil diameters during search tasks are smaller than those in nonsearch tasks. We postulate that this may be a result of the layout of the GUI, and thus future research should investigate whether this phenomena persists in other GUIs. If there is a difference in pupil diameter among the two conditions, implications with respect to operator workload should be investigated, and the reliability and robustness of this metric in the context of supervisory control should be further assessed.

Chapter 3

Robust Task-Scheduling Strategies for Multi-Operator Missions

The previous chapter demonstrated the investigation of operator behavior through *passive* observation and post-processing assessment. This information can be used to improve operator performance in an *offline* fashion by altering user interfaces and data presentation. However, information about operator cognitive behavior can also be used to develop decision support systems that seek to improve system performance in an *online* fashion, i.e., as the mission progresses and new information becomes available.

This chapter illustrates a particular operator-focused coordination scheme that seeks to improve human performance (and thus system performance) by optimizing the operator's *schedule*, i.e., the order and time at which tasks are processed. This approach is applicable to any supervisory application requiring sequential task-processing by an operator or team of operators, e.g., human analysis of data sets generated by autonomous mobile sensors. In particular, this type of scheduling is relevant to many surveillance tasks that rely on operator analysis of camera imagery to find or monitor targets.

The following analysis outlines a heuristic scheme to generate operator schedules in

a manner that both takes into account human cognitive requirements and is robust to various types of behavioral and modeling uncertainty. For full generality, we present the scheme assuming the possibility of multiple operators, though the scheme readily applies to the single operator case.

3.1 Multiple Operator Scheduling

3.1.1 Scheduling Objective

Suppose there are $M \in \mathbb{N}$ heterogenous tasks stacked in a queue awaiting the attention of any one of $L \in \mathbb{N}$ operators. We assume that each task T_j, where $j \in \{1, \ldots, M\}$, has an associated processing time $\tau_j \in \mathbb{R}_{>0}$, which defines how long the task will take to complete, an availability time $A_j \in \mathbb{R}_{>0}$, which defines the global time at which the task becomes available for processing, and an associated reward $R_j \in \mathbb{R}_{\geq 0}$, which is awarded upon successful completion. Assume, for the moment, that τ_j is known *a priori* (we relax this assumption later). Also assume that the parameters τ_j, A_j, and R_j are independent of which operator processes the task. Further, assume that all operators are aware of which tasks are available, and which tasks have already been processed at any time.

For the purposes of this study, we consider an "all or nothing" reward distribution scheme, in which an operator receives the full reward R_j if the task is completed, and no reward otherwise. Further, the reward for a task is only obtained if some operator completes the task within a pre-specified time horizon $T_H \in \mathbb{R}_{>0}$.

With this framework, we seek an optimal multi-operator schedule, i.e., ordered sequence of tasks for each operator, that maximizes the total reward accumulated across all operators. Note that, given any ordered sequence of tasks for a single operator, one can easily define appropriate starting times for each element of the sequence so that the

availability time constraints A_j are not violated. Indeed, the time at which the operator should start each task in the sequence can be taken as the maximum of the completion time of the previous task in the sequence, and the appropriate start time constraint.

We formally characterize the nominal problem statement as follows: define a *(multi-operator) schedule* as a set of L sequences $\{S_\ell \coloneqq \{T_{\sigma(1,\ell)}, T_{\sigma(2,\ell)}, \ldots, T_{\sigma(M_\ell,\ell)}\}\}_{\ell\in\{1,\ldots,L\}}$, where $\sum_{\ell=1}^{L} M_\ell \leq M$ and $\sigma : \sqcup_{\ell=1}^{L}\{1,\ldots,M_\ell\} \to \{1,\ldots,M\}$ is some injective mapping (here, $\sqcup\cdot$ is the set theoretic disjoint union). Let $\mathcal{S}$ denote the set of all possible schedules. We seek a schedule $S^* \in \mathcal{S}$, so that if each operator ℓ were to start the first task in the sequence at time $A_{\sigma(1,\ell)}$, and the k-th task in the sequence at time $\max\{C_{k-1,\ell}, A_{\sigma(k,\ell)}\}$ where $k \in \{2,\ldots,M_\ell\}$ and $C_{k-1,\ell}$ represents the time at which the $k-1$-st task in operator ℓ's sequence is completed , then the total accumulated reward across all operators over the time horizon T_H is maximized.

3.1.2 Incorporating Task Load Constraints

Since we consider the scheduling problem primarily in the context of human supervisory systems involving UAVs, the effectiveness of a given schedule depends upon the cognitive states of the operators. Therefore, it is desirable to construct a schedule that allows each operator to maintain their cognitive state in a high-performance regime. In particular, we focus our attention on moderating the relationship between the chosen schedule and the resulting operator *task loads* (see Section 1.1), and thus we extend the nominal problem statement to incorporate additional load constraints.

Although there are many different ways of modeling task load, e.g., utilization ratio [141], multi-dimensional load space abstractions [142], among others, we model task load via a simple, incremental, discrete process, which is driven by the task processing order and task processing times. Our chosen model is based on the simple observation

that, in most situations, when operators are executing tasks, the imposed task load level increases, and when operators are idle, the imposed task load decreases. In order to capture this simple dynamic evolution, to each task T_j, we associate a task load increment ΔW_j, which represents the amount that the operator's task load increases from working on the task for time τ_j. Further, we assume that task load decreases by a certain amount when the operator is idle, proportional to idle time. We wish to solve the scheduling problem under the additional constraint that, if possible, each operator's task load should remain within the regime $[\underline{W}, \overline{W}] \subset \mathbb{R}$ at any time within the interval $[0, T_H]$.

Although this model may be simplistic, it captures the essence of task load evolution during sequential processing. Indeed, many widely accepted task load evolution models are deterministic processes that augment task load levels during busy times and degrade task load during idle times (e.g., [141]). Later, we will treat processing times as predetermined parameters for the optimization (even in the scenario-based formulation of Section 3.1.4). Therefore, if desired, the task load increment parameters can be systematically chosen to reflect more sophisticated dynamics. For example, if $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a function that relates time to the amount that task load increments when the operator is engaging in a general task, then we can set $\Delta W_j = f(\tau_j)$ for all $j \in \{1, \ldots, M\}$. A similar statement can be made regarding task load decrements during idle time.

Note that while there are numerous task load (and cognitive workload) models in the literature, many rely on large amounts of pre-existing data or the presence of extensive physiological sensing to calibrate and precisely model the operator load. We have chosen a simpler path that captures the main qualitative features of such activities. As part of the development of these types of approaches, it would be important to validate whether high level decision making problems (such as the one addressed in this chapter) could support such simplified models, or would necessitate more complex, higher fidelity models. Currently, this is an open question that we leave as a topic of future work.

Despite its flexibility, the incremental task load model considered here cannot account for dynamics in which increment or decrement magnitudes are dependent upon initial conditions, since the order in which tasks will be processed is unknown *a priori*. However, task load evolution is usually a subjective experience regardless, and thus fine level task load models may be ill-suited if they are derived from aggregate data. Therefore, in the construction of general schemes intended to be used with many different operators, simplistic dynamics may be preferable to finer level models. If, however, the model is meant to be tuned to a specific operator or group of operators, or more precise real-time data can be leveraged to accurately predict cognitive states (e.g., neurophysiological cues [143]), then alternative models may be preferable. In the latter case, the mixed-integer linear programming (MILP) framework presented herein may not be sufficient and other options should be explored.

3.1.3 Multi-Operator Scheduling as a MILP

We now illustrate how the finite horizon scheduling problem can be formulated as a MILP. For the time being, we consider here the deterministic case in which all task processing times τ_j are known exactly. In the MILP formulation, the primary decision variables are binary indicators $x_{j,k,\ell} \in \{0, 1\}$, which specify whether or not task T_j should be executed in the k-th time slot of operator ℓ's output schedule, i.e., task sequence (see Figure 3.1). In accordance with Sections 3.1.1 and 3.1.2, each task T_j is fully specified by the (fixed) 4-tuple $(\tau_j, A_j, R_j, \Delta W_j)$.

In addition to the tasks $\{T_1, \ldots, T_M\}$, we introduce one additional "null-task," T_{M+1}, to represent times during which the operator is idle. Specifically, we define the null task as $T_{M+1} \coloneqq (\zeta, 0, 0, -\Delta W_{M+1})$, where $\Delta W_{M+1} > 0$ is a constant and $\zeta \in \mathbb{R}_{\geq 0}$ is a parameter representing its length. With the goal of capturing task load evolution within

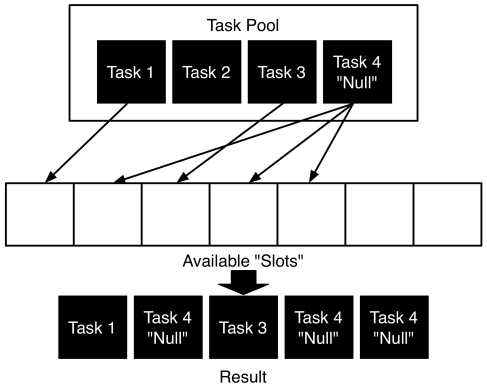

Figure 3.1: A diagram illustrating the basic MILP solution approach for an example with 4 tasks. The binary decision variables $x_{j,k,\ell}$ essentially "choose" tasks from the available task pool to fill available "slots" in each operator's schedule. Idle times are specified via the null task, T_{M+1}, which is the only task in the pool that can be chosen to fill multiple slots in an operator's output schedule.

the MILP framework, we augment the set of tasks to be processed with the newly created null task, and re-define $\mathcal{T} \coloneqq \{T_1, \ldots, T_M, T_{M+1}\}$.

The null-task is the only task that the operator may execute more than once. In other words, the output of our proposed method is an *augmented schedule*, which is formally defined as a set of sequences $\{S_\ell \coloneqq \{T_{\sigma(1,\ell)}, T_{\sigma(2,\ell)}, \ldots, T_{\sigma(M_\ell,\ell)}\}\}$, where each index $M_\ell \in \mathbb{N}$ is upper bounded by a fixed parameter $K \in \mathbb{N}$, and $\sigma : \sqcup_{\ell=1}^{L}\{1, \ldots, M_\ell\} \to \{1, \ldots, M+1\}$ is some mapping such that the pre-image of each singleton set $\{j\}$, where $j \in \{1, \ldots, M\}$, contains at most 1 element. This differs from the definition of *schedule* in that it allows the possibility of the null-task appearing more than once, and thus the parameter K, which represents the maximum number of tasks that can appear in any single operator's sequence, may exceed M. To guarantee reasonable output augmented schedules, it is necessary to pick K sufficiently large, i.e., to consider output sequences with a sufficiently large number of terms. Setting $K \geq T_H/\zeta + M$ is sufficient for our purposes. For the remainder of this chapter, when the distinction between a schedule and an augmented schedule is not of particular relevance, we simply say "schedule," and reserve the qualifier "augmented" only for cases where the distinction is consequential.

With this structure, we can incorporate task load into a MILP as an explicit variable that satisfies an appropriate set of constraints. As such, the scheduling problem with the additional task load consideration reduces to that of finding the optimal augmented schedule based on the set $\mathcal{T}$. We rigorously develop this formulation here.

Feasibility Constraints: We first pose a set of constraints to ensure that the solution to the MILP corresponds to a feasible solution to the scheduling problem. Consider the following constraints on the binary decision variables $x_{j,k,\ell}$:

$$\sum_{j=1}^{M+1} x_{j,k,\ell} \leq 1, \qquad \forall k \in \{1,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}, \tag{3.1}$$

$$\sum_{\ell=1}^{L}\sum_{k=1}^{K} x_{j,k,\ell} \leq 1, \qquad \forall j \in \{1,\ldots,M\}, \tag{3.2}$$

$$\sum_{j=1}^{M+1} x_{j,k,\ell} - x_{j,k-1,\ell} \leq 0, \qquad \forall k \in \{2,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}. \tag{3.3}$$

The constraint (3.1) specifies that each time slot, with respect to each operator's sequence order, can contain at most one task. Similarly, the constraint (3.2) guarantees that each task can only be assigned to at most a single location in the sequence, with the exception of the null task (note the constraint does not include the null-task). The constraint (3.3) guarantees that each operator's task "slots" are filled successively. That is, if some operator's task slot is filled with a task, then all of the previous slots must be filled as well. Together, these three constraints ensure that the output solutions correspond to valid augmented schedules, according to the rigorous definition given.

Availability, Start Time, and Completion Time Constraints: The next constraint set defines appropriate choice of task start times and completion times. Define $B_{k,\ell}$ and

$C_{k,\ell}$ as non-integer decision variables that denote the start time and the completion time of the k-th task in operator ℓ's sequence, resp. Consider the following constraints:

$$\sum_{j=1}^{M+1} x_{j,k,\ell} A_j \leq B_{k,\ell}, \qquad \forall k \in \{1,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}, \tag{3.4}$$

$$\sum_{j=1}^{M+1} x_{j,k,\ell} \tau_j = C_{k,\ell} - B_{k,\ell}, \qquad \forall k \in \{1,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}, \tag{3.5}$$

$$C_{k,\ell} \leq T_H, \qquad \forall k \in \{1,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}, \tag{3.6}$$

$$0 \leq B_{1,\ell} \leq \zeta, \qquad \forall \ell \in \{1,\ldots,L\}, \tag{3.7}$$

$$0 \leq B_{k,\ell} - C_{k-1,\ell} \leq \zeta. \qquad \forall k \in \{2,\ldots,K\}, \forall \ell \in \{1,\ldots,L\}, \tag{3.8}$$

The constraint (3.4) guarantees that no task is started before the specified availability time. The constraint (3.5) guarantees that start times and completion times are related correctly. The constraint (3.6) specifies that rewards are only attained for those tasks that are completed within the time horizon T_H. The contributions of constraints (3.7) and (3.8) are two-fold: First, the lower bounds specify that, in any individual operator's sequence, no task can begin before the previous task ends. The upper bounds arise from the fact that we have discretized operator idle time into discrete sets of length ζ. These constraints limit "gaps" that may be present in the output schedule (see Remark 1).

Task Load Evolution Constraints: The remaining constraints deal with moderating task load. Define $W_{k,\ell}$ as a non-integer decision variable indicating operator ℓ's task load level after the k-th task in their sequence, and let β, γ be additional, auxiliary

non-integer decision variables. Consider the following constraints:

$$W_{0,\ell} + \sum_{j=1}^{M+1} \Delta W_j x_{j,1,\ell} = W_{1,\ell}, \qquad \forall \ell \in \{1, \ldots, L\}, \tag{3.9}$$

$$W_{k-1,\ell} + \sum_{j=1}^{M+1} \Delta W_j x_{j,k,\ell} = W_{k,\ell}, \qquad \forall k \in \{2, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \tag{3.10}$$

$$W_{k,\ell} - \overline{W} \leq \beta, \qquad \forall k \in \{1, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \tag{3.11}$$

$$\underline{W} - W_{k,\ell} \leq \gamma, \qquad \forall k \in \{1, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \tag{3.12}$$

$$\beta \geq 0, \tag{3.13}$$

$$\gamma \geq 0. \tag{3.14}$$

With our incremental definition, it is necessary to ensure that operator ℓ's task load level, $W_{k,\ell}$, after the k-th task is defined precisely as the task load $W_{k-1,\ell}$ after execution of the previous task plus the increment defined in the task definition. This is captured by the constraints (3.9) and (3.10). Here, $W_{0,\ell}$ denotes operator ℓ's initial task load. Notice that, since we have included the null-task in the pool of available tasks, this set of constraints also defines the appropriate change in task load due to idle time. The constraints (3.11), (3.12), (3.13), and (3.14) state that the task load levels need to remain within the pre-specified bounds, buffered by the decision variables β and γ. These auxiliary "buffer" variables will factor into a linear objective function to penalize schedules that exceed the specified bounds, as discussed in the subsequent section.

MILP Formulation: With these constraints in place, the full MILP takes the form (3.15). We note that this formulation is a natural extension of that in our previous work [144],

which considers an analogous problem for a single operator scenario.

$$\begin{aligned} \text{Maximize:} \quad & \left(\sum_{j=1}^{M} \sum_{k=1}^{K} \sum_{\ell=1}^{L} R_j x_{j,k,\ell} \right) - p_\beta \beta - p_\gamma \gamma \\ \text{Subject To:} \quad & (3.1) - (3.14) \end{aligned} \tag{3.15}$$

Here, the optimization is performed over the binary variables $x_{j,k,\ell}$ and the non-integer variables $B_{k,\ell}, C_{k,\ell}, W_{k,\ell}, \beta$ and γ. The remaining variables are fixed parameters. Notice that the buffer variables β and γ enter into the objective function as linear penalties for violating the task load constraints, proportional to the parameters $p_\beta, p_\gamma > 0$. Enforcing the task load bounds in this manner, rather than as a strict constraint, is beneficial for a variety of reasons. First, exact bounds $\underline{W}, \overline{W}$ are usually not known beforehand, and thus enforcement of a strict bound may be ill-advised. Second, the variables p_β and p_γ provide the system designer with a means to tune the degree of enforcement of task load bounds. Indeed, higher values of p_β and p_γ lead to higher penalties in the objective function for bound violations. Finally, the enforcement of task load penalties as a soft constraint ensures feasibility of at least 1 non-degenerate solution, assuming that the time horizon T_H is sufficiently long. This leads to the following observation.

Lemma 1 (Feasibility) *The optimization* (3.15) *is feasible. If, in addition, there exists* $j \in \{1, \ldots, M\}$ *such that* $\tau_j + A_j < T_H$*, then there exists a non-degenerate feasible point, i.e., a choice of decision variables for which there is at least* 1 *task in the associated output augmented schedule.*

Proof: The zero vector is feasible, implying nominal feasibility. First assume that there exists $j \in \{1, \ldots, M\}$ such that $A_j = 0$ and $\tau_j < T_H$. In this case, we can construct a feasible decision vector by choosing $x_{j,1,1} = 1$, all other binary decision variables are equal to zero, $B_{1,1} = 0$, $C_{1,1} = \tau_j$, and $\gamma = \beta = C$, where C is a very large constant (bounding

workload violations). Simple substitution verifies feasibility. When there exists j such that $A_j + \tau_j < T_H$ and $A_j \neq 0$, we can construct a feasible decision vector using a similar procedure, inserting null-tasks at the beginning of the schedule as necessary to ensure that (3.7) is satisfied and re-defining the non-integer variables accordingly. ■

Most formulations of the optimal scheduling problem are known to be NP-hard. The MILP (3.15) presents a similar set of difficulties. Despite difficulties in finding global optima for all problem types, effective heuristics and reasonable methods exist for finding high quality solutions to MILPs [145]. Such methods include rounding schemes, branch and bound search strategies, genetic algorithms, simulated annealing schemes, and many others [146, 147]. Efficient implementations of many such solution heuristics are included in a variety of software packages, including the Matlab optimization toolbox [148], GLPK [149], and cvx [150, 151]. Therefore, if strict global optima are not of primary concern, the MILP formulation (3.15) may provide a viable solution.

Remark 1 (Null-Task Granularity) *Recall that* (3.15) *relies on the creation of a discrete "null-task" with length* ζ*. Once a solution to the MILP is found and an augmented schedule is extracted, tasks are sequentially executed by each operator. Due to task availability constraints, however, it may not be possible for the operator to start a new task immediately after the previous task is completed. Further, the solver may also introduce artificial delays between successive null-tasks. Therefore, if the time-profile of the operator's task execution is mapped over the horizon* T_H*, there may be "gaps", i.e., times in which no task (even a null-task) is executed. Task load effects during these gaps are not explicitly taken into account in* (3.15)*. However, a result of* (3.7) *and* (3.8) *is that the length of any possible gaps shrink to* 0 *as* $\zeta \to 0$*. Therefore, if* ζ *is taken small enough, then task load effects due to unaccounted gaps become negligible. Of course, shrinking* ζ *increases computational complexity, and thus there is a tradeoff that must be addressed.*

3.1.4 Incorporating Uncertainty

A major drawback of the scheduling formulation of Section 3.1.3 is that it does not directly incorporate uncertainty in system parameters. As mentioned, uncertainty is inherent to mixed human-machine teaming applications, and steps should be taken to design a system that takes this issue into account. In response, we focus our attention in this section on designing scheduling schemes that are robust to uncertainty in task processing times, which is usually a significant source of uncertainty in persistent task analysis missions. As such, assume that, for each T_j, the processing time τ_j is a random variable, which is distributed according to a probability density function f_j. We assume for now that each distribution f_j is known with complete certainty *a priori*. For simplicity, we assume that the distributions f_j are operator independent; however, we note that, if desired, operator-dependent processing time distributions can be added to our formulation through straightforward extension (see Remark 5). Generally, the choice of appropriate function f_j is dependent upon the nature of the task being processed. However, in some circumstances, standard distributions, such as log-normal distributions, have been shown to be effective in the context of collaborative human-UAV missions [32, 152].

We adopt a scenario-based approach to addressing uncertainty. According to this approach, the processing time distributions of each task are sampled to generate a set of *scenarios*, or possible task processing times. These scenarios are then used to find an augmented schedule, which remains feasible regardless of the scenario that is chosen to represent the true task processing times. If the underlying distributions are accurate, then using large numbers of scenarios to generate a task processing schedule enables the result to be robust to particular realizations of the processing time variables in actuality. That is, by following the schedule obtained, a wide variety of task processing times will still produce rewards that are above the predicted lower bound. Note that the optimal

choice for the number of scenarios will generally be application dependent. A thorough study of the optimal number of scenarios with respect to a given performance metric is an open research problem that we do not consider here. For our purposes, we choose values for this parameter that effectively illustrate qualitative effects on the resultant solution.

More formally, suppose that, for each task T_j, we generate a set of $Q \in \mathbb{N}$ possible processing times, $(\tau_j^1, \tau_j^2, \ldots, \tau_j^Q)$, where $\tau_j^q \sim f_j$ for all q. For each $q \in \{1, \ldots, Q\}$, the set $\{\tau_1^q, \ldots, \tau_M^q, \zeta\}$ defines a *scenario*, since it represents a possible realization of task processing times (note that the null task is included). Given the set of Q scenarios, we expand the MILP (3.15) through the additional requirement that any constraint relating to availability, start times, completion times, or task loads must be satisfied for *all* of the generated scenarios. Amendments to the relevant constraints to satisfy this additional robustness requirement are summarized as follows.

Availability, Start Time, and Completion Time Constraints: Define $B_{k,\ell}^q$ and $C_{k,\ell}^q$ as non-integer decision variable that denote the start time and the completion time of the k-th task in operator ℓ's sequence according to the q-th scenario, respectively. We also introduce one additional auxiliary, non-integer decision variable α. We enforce the following constraints (which are analogous to (3.4) - (3.8)):

$$\sum_{j=1}^{M+1} x_{j,k,\ell} A_j \leq B_{k,\ell}^q, \quad \forall k \in \{1, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \forall q \in \{1, \ldots, Q\} \quad (3.16)$$

$$B_{k,\ell}^q + \sum_{j=1}^{M+1} x_{j,k,\ell} \tau_j^q = C_{k,\ell}^q, \quad \forall k \in \{1, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \forall q \in \{1, \ldots, Q\} \quad (3.17)$$

$$C_{k,\ell}^q - T_H \leq \alpha, \quad \forall k \in \{1, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \forall q \in \{1, \ldots, Q\} \quad (3.18)$$

$$0 \leq B_{1,\ell}^q \leq \zeta, \quad \forall \ell \in \{1, \ldots, L\}, \forall q \in \{1, \ldots, Q\} \quad (3.19)$$

$$0 \leq B_{k,\ell}^q - C_{k-1,\ell}^q \leq \zeta. \quad \forall k \in \{2, \ldots, K\}, \forall \ell \in \{1, \ldots, L\}, \forall q \in \{1, \ldots, Q\}. \quad (3.20)$$

Task Load Evolution Constraints: Define $W^q_{k,\ell}$ as a non-integer decision variable indicating operator ℓ's task load level after the k-th task in their sequence according to the q-th scenario. Define all other variables as before. Consider the following constraints:

$$W_{0,\ell} + \sum_{j=1}^{M+1} \Delta W_j x_{j,1,\ell} = W^q_{1,\ell}, \qquad \forall \ell \in \{1,\dots,L\}, \forall q \in \{1,\dots,Q\}, \tag{3.21}$$

$$W^q_{k-1,\ell} + \sum_{j=1}^{M+1} \Delta W_j x_{j,k,\ell} = W^q_{k,\ell}, \qquad \forall k \in \{2,\dots,K\}, \forall \ell \in \{1,\dots,L\}, q \in \{1,\dots,Q\} \tag{3.22}$$

$$W^q_{k,\ell} - \overline{W} \leq \beta, \qquad \forall k \in \{1,\dots,K\}, \forall \ell \in \{1,\dots,L\}, \forall q \in \{1,\dots,Q\} \tag{3.23}$$

$$\underline{W} - W^q_{k,\ell} \leq \gamma, \qquad \forall k \in \{1,\dots,K\}, \forall \ell \in \{1,\dots,L\}, \forall q \in \{1,\dots,Q\} \tag{3.24}$$

$$\alpha \geq 0, \tag{3.25}$$

$$\beta \geq 0, \tag{3.26}$$

$$\gamma \geq 0. \tag{3.27}$$

Note that the constraints (3.21) - (3.27) are analogous to (3.9) - (3.14). Also note that, if desired, we can easily make the workload increments scenario-dependent as well. In this case, ΔW_j in (3.21) and (3.22) would be replaced by a scenario-dependent parameter ΔW^q_j. This is often useful if workload increments are a function of the task processing time. We take this approach for the numerical simulations presented later in this chapter.

Scenario-Based MILP Formulation The scenario-based MILP is expressed in (3.28).

$$\begin{aligned} \text{Maximize:} \quad & \left(\sum_{j=1}^{M} \sum_{k=1}^{K} \sum_{\ell=1}^{L} R_j x_{j,k,\ell} \right) - p_\alpha \alpha - p_\beta \beta - p_\gamma \gamma \\ \text{Subject To:} \quad & (3.1) - (3.3) \\ & (3.16) - (3.27). \end{aligned} \tag{3.28}$$

Here, the optimization is performed over the binary variables $x_{j,k,\ell}$ and the non-integer variables $B^q_{k,\ell}, C^q_{k,\ell}, W^q_{k,\ell}, \alpha, \beta$ and γ. The remaining variables are fixed parameters. Note that (3.28) is largely the same as (3.15); however, there are a few key differences. First, for each q, there are unique sets of decision variables $\{B^q_{k,\ell}\}, \{C^q_{k,\ell}\}$, and $\{W^q_{k,\ell}\}$, where $k \in \{1, \ldots, K\}$ and $\ell \in \{1, \ldots, L\}$, yet there is only one set of binary variables $\{x_{j,k,\ell}\}$ which serves all scenarios. Thus, starting times, ending times, and task load values are scenario-dependent, while the resulting schedule is not dependent on these parameters. This allows for the unambiguous extraction of an augmented schedule from the solution. The result will then (ideally) be robust to uncertainty in processing times. Second, an additional decision variable α bounds the amount that the task completion times can exceed the horizon T_H. This variable α is then factored into the objective function by means of a linear penalty. As with task load, this "soft" enforcement of the time horizon constraint is advantageous because it provides system designers with an additional "tuning" parameter to control the degree of robustness. Indeed, by increasing the value of p_α, more penalty is incurred for generating schedules whose completion times are likely to exceed T_H. These additional parameters also serve to prevent the optimization from returning degenerate solutions in the case of highly skewed processing time distributions. This discussion readily leads to the following lemma, whose proof straightforward and thus omitted.

Lemma 2 *There always exists at least* 1 *non-degenerate solution to* (3.28).

Remark 2 (Robust Optimization) *Many schemes exist for solving optimization problems containing uncertainty. In particular, the area of robust optimization provides tools for finding solutions that perform well in the worst case, given that the underlying uncertainty sets are accurately bounded [70]. Scenario-based approaches to handling uncertainty can, in some sense, be viewed as a "naive" approach to robust optimization since they rely on a sampling-based expansion of the constraint set. Despite this, scenario-based approaches lend themselves well to a variety of applications due to their simplicity and effectiveness. We adopt a scenario-based approach for the following reasons: First, scenario-based optimization is simple and intuitive, making it an attractive to practitioners and theoreticians alike. Second, most other approaches to robust optimization rely on the presence of bounded uncertainties, or require approximations that enforce artificial bounds. The scenario-based approach does not require such bounds. Third, scenario-based approaches allow for easy tuning of the problem to fit with a desired degree of robustness. Finally, given "regularly" shaped uncertainty distributions, scenario-based approaches usually provide reasonable performance with a modest number of samples.*

Remark 3 (Scenario Generation) *Scenarios are generated by sampling the functions f_j. Naive Monte-Carlo sampling usually produces samples that most accurately reflect the underlying distribution in the limit as the number of samples grows. However, scenarios can be generated by any reasonable sampling method and, in certain situations, design goals may be better served by these alternative means. For example, processing time distributions may have semi-infinite support, and thus Monte-Carlo sampling can produce some scenarios with excessively long durations in comparison to the horizon length. In most cases, excessively long durations will lead to degenerate, or excessively conservative solutions. Therefore a more restrictive sampling scheme may be beneficial.*

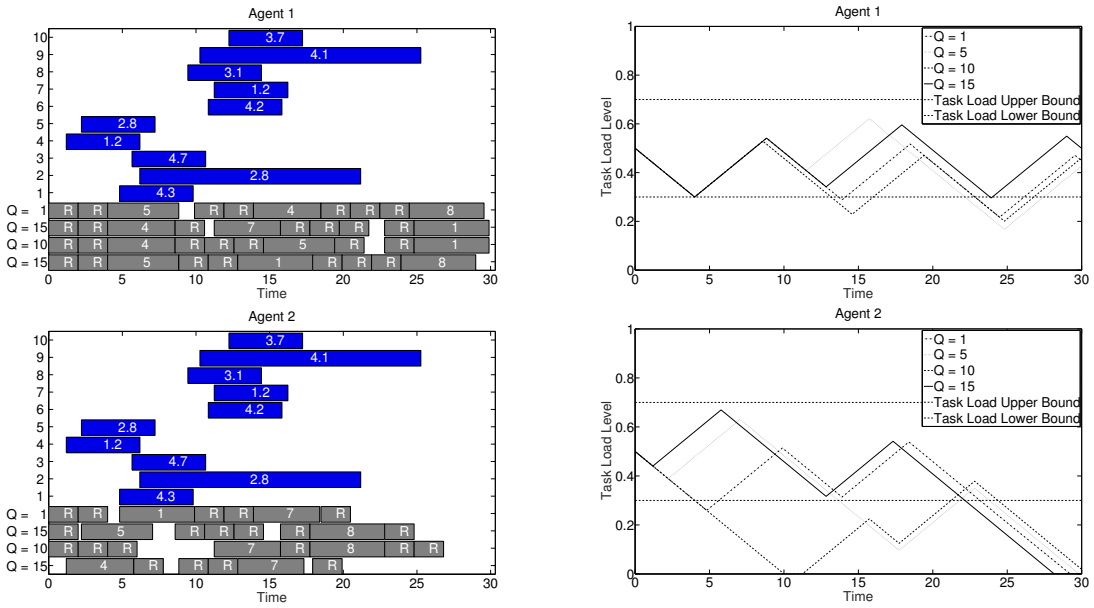

Figure 3.2: An example of a multi-operator (augmented) schedule produced by solving (3.28), for an example mission with 10 tasks, 2 operators (labeled "Agent 1","Agent 2"), a total horizon length of 30 (dimensionless units), and log-normally distributed task processing times.

3.1.5 Numerical Examples

Figure 3.2 shows a summary of scheduling solutions for an abstracted and simplified mission with 10 tasks, 2 operators, a horizon length of 30 (dimensionless units), and log-normally distributed task processing times. The solution was calculated using the formulation (3.28). For this simulation, we chose $p_\alpha = 10$, $p_\beta = p_\gamma = 15$, $\underline{W} = 0.3$, $\overline{W} = 0.7$, and $W_{0,1} = W_{0,2} = 0.5$. In the two plots in the left-most column of Figure 3.2, the problem setup is portrayed using the blue bars. For each task T_j, the appropriate bar starts at the task availability time A_j, and extends for a length corresponding to the expected processing time. The number inside the bar represents the reward obtained for successful completion of the task. The gray bars in the lower portion of the plot represent a simulated instance of the operator task execution for each scenario condition, based on the solution to (3.28). The start time and length of the bar represent the time that

the task was started and the task duration respectively, while the number inside the bar represents the task index, with a letter "R" indicating a null-task ("Rest"). Specifically, these simulated schedules were calculated as follows. First, "actual" task execution times, i.e., realizations of each task processing time, were randomly sampled from the appropriate distribution. Then, for each scenario condition, scenarios were generated using naive Monte-Carlo sampling (see Remark 3), and the optimization (3.28) was solved using Matlab's `Intlinprog` function. Using the resultant schedule, task execution was simulated by assuming that each operator processes each task in the respective sequence specified by the optimization solution and that each task has a duration according to the "actual" processing time. Each task was started at the earliest possible time (the maximum of the previous task completion time and the availability constraint), and tasks were only included if their actual completion time was less than T_H. Note that all task executions satisfy the availability constraints, as expected, but their actual duration differs from the expected duration due to uncertainty in processing times.

The two plots in the right-most column of Figure 3.2 show the evolution of task load corresponding to the simulated mission that is summarized by the corresponding plots on the left. In the simulation, a simple linear task load model is considered: if $t \in \mathbb{R}_{\geq 0}$, we assume operator ℓ's task load W_ℓ satisfies the dynamics:

$$\frac{\mathrm{d}W_\ell}{\mathrm{d}t}(t) = \begin{cases} 0.05, & \text{if executing a non-null task at time } t \text{ and } W(t) \neq 1, \\ -0.05, & \text{if executing a null task or no task at time } t \text{ and } W(t) \neq 0, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

We have chosen to constrain task load values to lie between 0 and 1 both for modeling purposes, and because this is a normalization that arises naturally in common task load models [153]. In *a priori* planning, this cap is not taken into account by the optimization,

and thus, for planning, it is assumed that $\Delta W_j^q := \pm 0.05\tau_j^q$, where the sign is dependent upon whether the task under consideration is a null-task. Note here, that as the number of scenarios increases, the task load does not exceed the specified bounds.

Figure 3.3 illustrates the effects of altering the task load bounds on achieved performance for a single-operator scheduling mission. The plot was generated as follows. First, an underlying task set was generated, the parameters $T_H = 30$, $W_{0,1} = 0.5$, and $p_\alpha = 10$ were initialized, and a set of 10 scenarios were generated. Then, 10 simulations runs were conducted, where in each run, (i) "actual" task processing times were sampled from the appropriate distribution, (ii) an augmented schedule was created using a solution to (3.28) for each set of $\underline{W}, \overline{W}, p_\beta$, and p_γ values (for the "no task load" condition, set these parameters to 0), (iii) the task execution process was simulated using the resulting schedule and "actual" processing times, and (iv) the achieved nominal reward was recorded, i.e., the sum of the R_i's for tasks that could be executed within time T_H. The top plot in Figure 3.3 shows the mean and standard deviation of the rewards obtained over the simulation runs (the absence of error bars indicates no variation across runs). The bottom two plots in Figure 3.3 show the maximum amounts that the task load exceeded the upper task load bound and the maximum amount that the task load violated the lower task load bound during task execution, respectively (not taking into account violations that occur after all tasks in an operator's schedule have been executed). Notice that the obtained reward decreases as p_β and p_γ are increased and as the allowable task load bounds widen, while the magnitude of task load bound violation decreases.

3.2 Adaptive Scheduling Scheme

The use of *a priori*, scenario-based robust scheduling strategies can generate reliable lower bounds on system performance. That is, using a reasonable number of scenarios

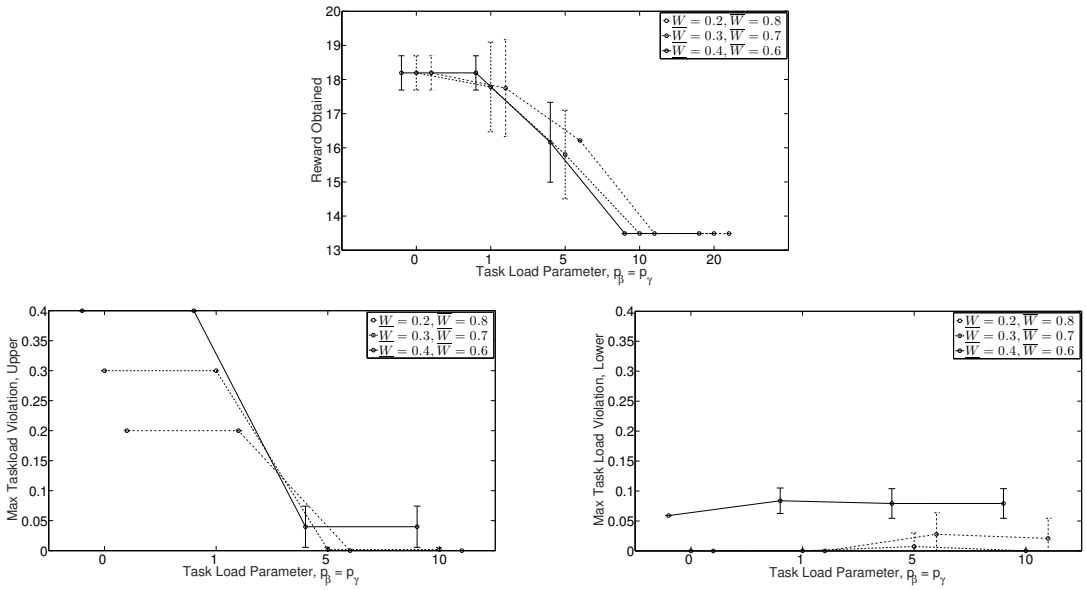

Figure 3.3: An illustration of the effects of altering the parameters $\underline{W}$, $\overline{W}$, p_β, and p_γ when using the optimization problem (3.28) to generate schedules for a single-operator sample mission.

(assuming that the chosen processing time distributions are accurate), the MILP (3.28) will produce a theoretical lower reward bound that will likely hold true in actuality. In missions where accurate performance guarantees are crucial, such bounds may be sufficient. However, theoretical bounds produced by scenario-based robust optimizations may be very conservative, particularly when the uncertainty distributions are highly skewed or have high variance. For example, if visual search times are modeled via log-normal distribution, then robust scheduling strategies will be most likely driven by search times occurring in the tail, which are unlikely to occur in actuality. As such, it is clear that naively following a robust schedule that is calculated *a priori* does not take advantage of the full knowledge at the designer's disposal during the mission, and thus may lead to poor solutions with respect to actual realizations of uncertain parameters.

In this section, we explore adaptive methods for improving performance while still maintaining the desired robustness properties of solutions. We start by developing strate-

gies for the single-operator case (i.e., the case where $K = 1$), and subsequently show how they can be extended for use with multiple operators.

3.2.1 Single Operator Receding-Horizon Scheduling

Assume a single operator is charged with sequentially processing a set of tasks. Performance under a robust scheduling strategy can potentially be improved by re-planning in real-time as task processing times are incrementally realized. This observation naturally leads to a receding-horizon scheme, which calculates new, robust schedules after each task is processed. This scheme is described in Algorithm 1.

Algorithm 1: *Single Operator Receding-Horizon Scheduling*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{f_j\}_{j\in\{1,\dots,M\}}$; $W_{0,1}$; Q, p_α, p_β, p_γ

while *Non-null tasks remain and the remaining horizon* $T_H > 0$ **do**

1. Collect the tasks to be processed in a set $\mathcal{T}$;
2. Formulate and solve the scenario-based robust scheduling problem as in Section 3.1.4, with respect to the set $\mathcal{T}$ and a horizon length T_H ;
3. **if** *Resultant schedule is empty or consists of only null tasks* **then** **break**
4. Operator executes the resultant schedule, until the first non-null task T_j is completed;
5. Remove T_j from $\mathcal{T}$, and observe the time t that has elapsed since the last re-plan;
6. Subtract t from all remaining (non-null) task availability constraints, redefine $T_H = T_H - t$;

The added benefit of the receding-horizon approach is that it takes advantage of the manner in which uncertainty arises in the sequential task analysis mission. Indeed, in these missions, uncertainty is generally reduced as time progresses. Accordingly, the receding horizon scheme re-plans each time an uncertain variable is realized, which, in the robust planning case, will generally lead to better performance. The obvious drawback to this strategy is that it requires a MILP to be solved after each non-null task is executed; thus, the total computational complexity is significantly increased. Therefore, there is a tradeoff between solution quality and computation time that must be addressed.

Remark 4 (Computation time) *Even though the receding-horizon robust scheduling*

scheme may be computationally intense in its raw form, there are many simple steps that can be taken to adjust the scheme and fit it into a variety of computational frameworks. The simplest amendment is to reduce the number of scenarios used, which reduces complexity at the expense of robustness. A slightly more sophisticated strategy would only re-plan when a certain criterion is met, as opposed to re-planning after every non-null task execution. For example, one could choose to re-plan only if the observed time on the executed task is significantly different from the worst-case processing time predicted by the generated scenarios. For the sake of brevity, a thorough treatment of these types of amendments is not included here, and is left as a topic of future work.

3.2.2 Single-Operator, Receding-Horizon Scheduling with Uncertainty Set Estimation

It has been assumed thus far that the probability distribution f_j of processing times for each task is known exactly. Indeed, samples are drawn assuming certain distributions in order to generate the scenarios that are used in the optimization (3.28). In many cases, however, the distributions themselves may not be known exactly, and thus may need to be estimated during the course of task execution. Of course, depending on problem assumptions, such estimation may not be helpful, e.g., if all tasks are assumed to have completely independent distributions f_j. However, given appropriate problem structure, online estimation of uncertain distributions can potentially further boost performance when used in conjunction with the receding-horizon approach of Section 3.2.1. To illustrate, we develop a common scenario that can benefit from such estimation here.

Assume once again the presence of a single operator. Building on the formulation of the previous sections, suppose each task T_j is generated by one of $P \in \mathbb{N}$ sources, e.g., different regions of interest, and each such task can be of $H \in \mathbb{N}$ possible types, e.g.,

easy, medium, or hard. Assuming independent sampling, suppose further that associated to each source $p \in \{1, \ldots, P\}$, there exists an unknown, static probability mass function $g_p : \{1, \ldots, H\} \to [0, 1]$ which captures the likelihood that a task originating from source p is of type h. That is, the probability of any single task that is generated by source p being of type h is $g_p(h)$. Finally, to each type h, associate an unknown, static probability density function $f_h : \mathbb{R}_{\geq 0} \to \mathbb{R}$, which captures the distribution of possible processing times. Note that the processing time distributions f_h are region independent. To summarize, with these additions, each task T_j consists of a 6-tuple $T_j = (\tau_j, A_j, R_j, \Delta W_j, \rho_j, \eta_j)$, where A_j, R_j, and ΔW_j are defined as before, $\rho_j \in \{1, \ldots, P\}$ represents the source which generated the task (known to the operator), $\eta_j \sim g_{\rho_j}$ is the task type (unknown to the operator), and $\tau_j \sim f_{\eta_j}$ is the processing time (unknown to the operator).

The true distributions associated with both sources and types are unknown to the operator. However, in this new setup, there are commonalities among the distributions that can be exploited through estimation. Thus, we implement estimation in conjunction with the receding horizon approach as follows. For each source p and each type h, let $\hat{g}_p$ and $\hat{f}_h$ denote estimates of g_p and f_h, resp. The basic idea behind this adaptive scheduling approach is to incrementally update the estimates $\hat{g}_p$ and $\hat{f}_h$ as new information becomes available, i.e., as the operator processes tasks. The updated estimates are then used in subsequent scheduling operations. This process is described by Algorithm 2

Line 2 in Algorithm 2 requires explanation. Indeed, the operator does not know the task type of any unprocessed task with complete certainty, and thus it is necessary to make a decision about which distribution should be sampled for the purpose of generating scenarios. This choice can be made in many different ways. For example, a logical choice is a "maximum likelihood" method, where each parameter τ_j^q is selected by first selecting $\eta_j^* \in \arg\max_h \hat{g}_{\rho_j}(h)$ and subsequently sampling the distribution $f_{\eta_j^*}$. That is, each scenario is generated by sampling from the processing time distribution corresponding to

Algorithm 2: *Single Operator Receding-Horizon Scheduling with Estimation*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{\hat{f}_h\}_{h\in\{1,\dots,H\}}$, $\{\hat{g}_p\}_{p\in\{1,\dots,P\}}$; $W_{0,1}$; p_α, Q, p_β, p_γ

while *Non-null tasks remain and the remaining horizon* $T_H > 0$ **do**

1. Collect the tasks to be processed in a set $\mathcal{T}$;
2. Generate new scenarios according to the estimates $\{\hat{g}_p\}_{p\in\{1,\dots,P\}}$ and $\{\hat{f}_h\}_{h\in\{1,\dots,H\}}$;
3. Formulate and solve the scheduling problem (3.28);
4. **if** *Resultant schedule is empty or consists of only null tasks* **then**
   break
5. Operator executes the resultant schedule, until the first non-null task $T_{\hat{j}}$ is completed;
6. Observe $\eta_{\hat{j}}$ and $\tau_{\hat{j}}$; Update the estimates $\hat{g}_{\rho_{\hat{j}}}$ and $\hat{f}_{\eta_{\hat{j}}}$;
7. Remove $T_{\hat{j}}$ from $\mathcal{T}$, and observe the time t that has elapsed since the last re-plan;
8. Subtract t from all remaining (non-null) task availability constraints, redefine $T_H = T_H - t$;

the most likely type for task T_j, according to $\hat{g}_{\rho_j}$. We exploit this "maximum likelihood" sampling process in our remaining simulations. Once the sampling method has been established, it remains to choose a process for updating the distributions $\hat{g}_p$ and $\hat{f}_h$. The appropriate update method will generally be governed by the problem assumptions. To illustrate the functionality of the receding-horizon robust scheduling scheme with estimation, we assume that each element in the set $\{f_h\}_{h\in\{1,\dots,H\}} \coloneqq (\mu_h, \sigma_h)$ is log-normally distributed, where μ_h, σ_h are the standard log-normal distribution parameters. We also assume that some previous information is available regarding each distribution g_p and f_h in the form of a set of previous samples, accumulated prior to the current scheduling mission. With this information, upon completion of task T_j, the appropriate distributions $\hat{g}_{\rho_h}$ and $\hat{f}_{\eta_j}$ are updated using standard maximum likelihood estimation.

Figure 3.4 presents a comparison between the various solution methods discussed thus far for a sample mission involving 10 tasks, 2 possible task sources, and 3 possible task types. Here, it is assumed that there is uncertainty in task processing times, as well as in the task distributions g_p and the distributions f_h. Each distribution g_p was generated randomly, and the distributions f_h were generated by creating a set of log-normal distributions, each with identical σ parameters, and whose medians (e^μ) were spaced equally across the interval $[0, 0.5T_H]$. It is assumed that 10 prior samples from each

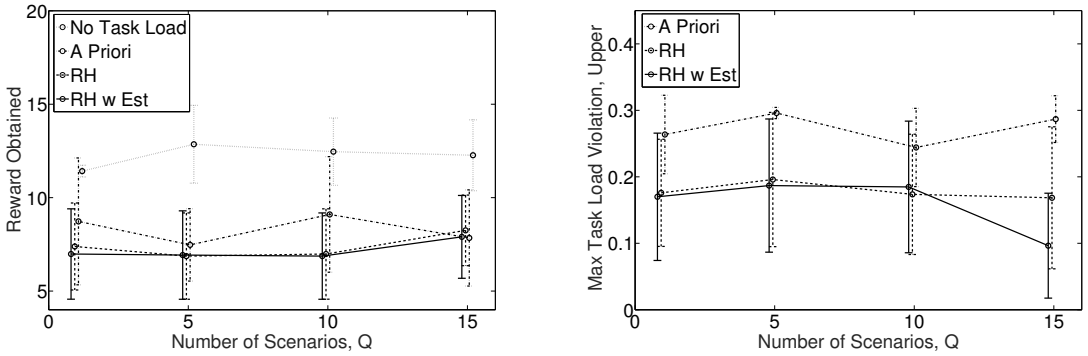

Figure 3.4: A performance comparison between 4 different solution methodologies: (i) *a priori* planning with no workload consideration, (ii) *a priori* planning with workload consideration, (iii) receding horizon planning with workload consideration, and (iv) receding horizon planning with estimation and workload considerations.

source distribution g_p are available *a priori*, and 5 prior samples are available from each distribution f_h. The prior samples were generated from sampling the appropriate "actual" distributions. The estimates $\hat{g}_p$ and $\hat{f}_h$ were then generated using standard maximum likelihood estimation. For the methods involving estimation, these distribution estimates were re-evaluated each time a task was completed. In all cases, scenarios were generated using the "maximum likelihood" scheme. Finally, we chose $p_\alpha = 10$, $p_\beta = p_\gamma = 15$, $W_0 = 0.5$, $\underline{W} = 0.3$, $\overline{W} = 0.7$, and $T_H = 30$. The left plot in Figure 3.4 shows the nominal rewards obtained, i.e., the sum of the the rewards R_i associated with tasks that were executed, averaged over 10 simulated task execution processes. The same setup and prior information was used in each run, but each run had different realizations of the processing times, sampled from the underlying distributions. Identical scenarios were used across experimental conditions (i.e., solution methods considered). As expected, the "no task load" condition resulted in the highest achieved rewards for all cases, and the rewards showed a slight downward trend as the number of scenarios increases, since higher numbers of scenarios provide increased robustness at the expense of lower expected rewards. Also note the large variances due to the high amounts of uncertainty in the

underlying problem. The right plot in Figure 3.4 shows the maximum amount that the upper task load bound was violated during the mission (the lower bound was almost never violated in any of the conditions, so the plot is omitted). Notice that, qualitatively, the receding horizon scheme with estimation resulted in lower task load violations than the other methods, while still achieving similar nominal rewards, particularly as the number of scenarios increased. Although the obtained benefit is quantitatively inconclusive in this example, we hypothesize that the adaptive scheme with estimation would likely produce a more pronounced benefit in situations with larger numbers of tasks. Rigorous analysis of this hypothesis and other effects due to the underlying problem structure on the utility of adaptive schemes is another interesting avenue of future research.

3.2.3 Multiple-Operator Receding-Horizon Schemes

On the surface, it may seem straightforward to extend the receding-horizon schemes of the previous two sections for use with multiple operators. However, upon closer examination, certain aspects of the previous algorithms become unclear in the presence of multiple operators that process tasks simultaneously. For example, in calculating schedules in the receding horizon framework for a single operator (Section 3.2.1), it is clear when re-planning operations are appropriate, e.g., when the operator finishes processing a task. However, with multiple operators, each operator will finish their assigned tasks at different times, and thus a decision must be made as to when it is appropriate re-plan. We explore these and other issues relating to multiple operator adaptive schemes here.

Assume the presence of $L > 1$ operators and the task structure of Section 3.2.2. As before, we assume that all task-related quantities are operator independent. That is, each task T_j is defined as before, with the parameters $\tau_j, A_j, R_j, \Delta W_j, \rho_j, \eta_j$ being independent of which operator processes the task. Further, the distributions g_p and f_h are

operator independent. Under these assumptions, the formulation (3.28) is appropriate for *a priori* planning of schedules. However, to develop a receding horizon scheme for multiple operators (analogous to Sections 3.2.1 and 3.2.2), some additional work is necessary. In particular, different operators will process and finish tasks at different times, and therefore an appropriate re-plan strategy must be formulated. In order to keep a close connection to the single cycle case, we develop a strategy in which we re-plan each time a non-null task is completed. This process is outlined in Algorithm 3.

Algorithm 3: *Multi-Operator Receding-Horizon Scheduling*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{\hat{f}_h\}_{h\in\{1,\dots,H\}}$, $\{\hat{g}_p\}_{p\in\{1,\dots,P\}}$; $\{W_{0,\ell}\}_{\ell\in\{1,\dots,L\}}$; Q, p_α, p_β, p_γ

while *Non-null tasks remain and the remaining horizon* $T_H > 0$ **do**

1. Collect the tasks to be processed in a set $\mathcal{T}$;
2. Call Algorithm 4 to formulate and solve a constrained multiple operator scheduling problem, in which tasks already in progress must be completed first by the appropriate operator;
3. **if** *Resultant (multi-operator) schedule is empty or consists of only null tasks* **then** **break**
4. Operators execute the new schedule until someone completes a non-null task $T_{\hat{\jmath}}$ (if an operator has a task in progress, this task will be the first task in their new schedule);
5. If desired, observe $\tau_{\hat{\jmath}}$, $\eta_{\hat{\jmath}}$ and update estimates $\hat{g}_{\rho_{\hat{\jmath}}}$ and $\hat{f}_{\eta_{\hat{\jmath}}}$;
6. Remove $T_{\hat{\jmath}}$ from $\mathcal{T}$, and observe the time t that has elapsed since the last re-plan;
7. Subtract t from all remaining (non-null) task availability constraints, redefine $T_H = T_H - t$;

Notice that line 2 in Algorithm 3 constructs new multiple operator schedule while simultaneously constraining the tasks already in progress to be completed first. This is done by adding some additional constraints to (3.28) before it is solved. For example, if task T_j is in progress by operator ℓ at the time of re-plan, then the additional constraint to be added would take the form $x_{j,1,\ell} = 1$. Indeed, setting $x_{j,1,\ell} = 1$ indicates that the first task in operator ℓ's new task sequence is task T_j. Recall, however, that upon re-plan, the actual processing time of any task in progress is still unknown. Therefore, in order for the MILP to accurately reflect the current mission state, prior to solving the MILP (3.28) (with the additional constraint just mentioned), it is also necessary to subtract the time that each task has been in progress from the generated scenarios. In

summary, for the previous algorithm, line 2 will require the steps in Algorithm 4.

Algorithm 4: *Constrained MILP Formulation*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{\hat{f}_h\}_{h\in\{1,\dots,H\}}$, $\{\hat{g}_p\}_{p\in\{1,\dots,P\}}$; $\{W_{0,\ell}\}_{\ell\in\{1,\dots,L\}}$; Q, p_α, p_β, p_γ
Output : Multi-operator (augmented) schedule

1 Generate new scenarios for the remaining tasks in $\mathcal{T}$;
for *Each task T_j in progress* **do**
2 Find the difference $\hat{t}$ between the current time and the time task T_j was started;
3 For all q, set $\tau_j^q = \max\{\tau_j^q - \hat{t}, 0\}$;
4 Find the agent ℓ that is working on task j. Add an additional constraint $x_{j,1,\ell} = 1$ to (3.28).
5 Formulate and solve (3.28), including the newly added constraints.;
return Multi-operator (augmented) schedule

Remark 5 (Generality) *Although we have made a variety of assumptions in formulating* (3.28)*, further generality can be achieved by altering the MILP in a natural way. For example, task processing times can be made operator dependent by generating scenarios for each operator, and introducing an additional index ℓ into the task processing time. The variables τ_j^q would then become $\tau_{j,\ell}^q$ to represent the time required for operator ℓ to process task T_j in the q-th scenario. Other generalizations can be made similarly. These generalizations are, of course, at the expense of additional computation.*

Remark 6 (Re-plan Schemes) *In the presented receding horizon scheme for multiple operators, we have chosen to re-plan each time a non-null task is completed. This is certainly not the only strategy. For example, an alternative strategy would re-plan only if a non-null task is finished and the actual task duration is significantly different than expected. Exploration of different re-plan strategies should be assessed in future work.*

Remark 7 (Task Pool Evolution) *A natural question that arises from the presented multiple-operator, receding horizon strategy is whether it is possible to simply omit tasks that are already in progress from the re-plan MILP, rather than altering scenarios and*

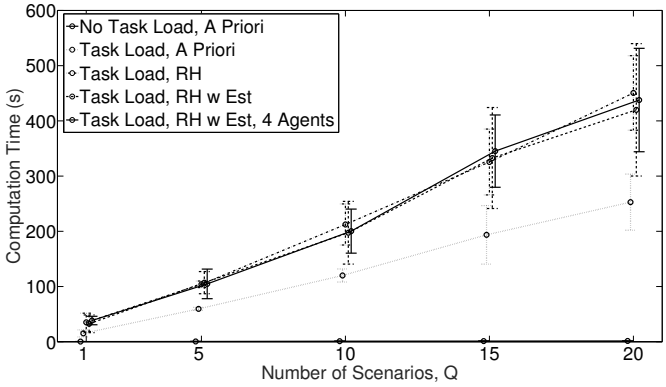

Figure 3.5: Observed computation times for different methodologies: (i) single agent, *a priori* planning, no task load consideration, (ii) single agent, *a priori* planning, task load consideration, (iii) single agent, receding horizon planning, task load consideration, (iv) single agent, receding horizon planning with estimation, task load consideration, and (v) 4 agents, receding horizon planning with estimation, task load consideration.

including additional constraints. This alternative strategy is not possible given the presented MILP framework, as the actual task end-times are unknown at the time of re-plan.

3.3 Heuristic Approach

The multiple operator formulation of Section 3.2.3 may be appropriate for small scheduling problems; however, the introduction of additional variables, which are necessary to move from a single to a multiple operator framework, and the additional planning instances that are required by adaptive approaches can significantly increase computation time, as shown in Figure 3.5. This figure was generated for an example problem involving $M = 10$ tasks, $T_H = 30$, $\zeta = 2$, $\underline{W} = 0.3$, $\overline{W} = 0.7$, $W_{0,\ell} = 0.3$ for all ℓ, $p_\alpha = 10$, and $p_\beta = p_\gamma = 15$. Note that even with a moderate number of tasks and operators, the inflation of the state space caused by the introduction of new variables and adaptive strategies significantly increases computation time. We remark that the plot

in Figure 3.5 is only meant to give the reader a flavor for the usual trends with respect to computation time, and is not meant to portray a strict relationship to be observed in every problem instance. Indeed, due to the use of heuristic solvers (namely, Matlab's `intlinprog` solver), observed computation times rely heavily on the particular problem setup and the parameters used to initialize solvers. For instance, there may be cases in which moving from a single agent setup to a 2 agent setup actually *reduces* computation time due to the nature of the state space. Despite this, the general trend remains, in that computation time diverges quickly to an impractical level as the number of agents grows and additional problem complexities are added.

This discussion motivates the need for alternative means for solving the multiple operator problem. In this section, we introduce one such method, and illustrate how this alternative method can still achieve adequate performance, while using only a fraction of the computation time that is observed when solving the problem with the naive approaches of Sections 3.1.4 and 3.2.3 . The idea behind this alternative approach is straightforward: instead of solving a single optimization across all operators, we use a heuristic to first assign tasks to operators, and subsequently solve the resulting single operator problems. In this section, we restrict our attention to *a priori* planning, although one could easily construct an analogous approach for receding-horizon implementation.

Consider first the case of a single scenario, i.e., $Q = 1$. Our proposed assignment strategy proceeds in a methodical fashion, in which operators are cyclically selected one at a time according to a pre-defined order. When an operator is selected, a single task is assigned to the selected operator out of the remaining task pool, based on scores that are assigned to each task. This procedure evolves according to Algorithm 5. In this algorithm, the pool of available tasks includes the null-task. The idea behind Algorithm 5 is to quickly and incrementally simulate a task-execution process for each operator, based on local information, and use the result as a basis for task assignment. Each task is assigned

a score based on how much the objective function of (3.28) would increment if that task were to be executed next in the considered operator's task sequence. The score for each task is discounted by a factor $\lambda^{t_j^{\text{end}}-\theta_j}$, which is proportional to the amount of time that it will take the operator to process the task (assuming that the time indicated by the scenario is accurate). The task with maximum score is assigned to the operator under consideration, and this task is "executed" next in the operator's simulated schedule.

Algorithm 5: *Single Scenario Task Assignment*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{W_{0,\ell}\}_{\ell\in\{1,\dots,L\}}$; p_α, p_β, p_γ
Output : Operator/Task Pairings

Initialize statistics $\alpha = \beta = \gamma = 0$, select $\lambda \in (0, 1]$;
for *Each Operator* ℓ **do**
  Initialize statistics $\theta_\ell = 0$, $\xi_\ell = W_{0,\ell}$
Select a permutation of the set $\{1, \dots, L\}$ to act as the selection order;
while *Unassigned non-null tasks remain (cycling through the selection order)* **do**
  for *The next operator in the selection order, say operator* ℓ **do**
    for *Each task* T_j **do**
      Calculate theoretical time t_j^{end} that operator ℓ could finish T_j, assuming θ_ℓ is the current time and the time prescribed by the scenario is the true processing time;
      Calculate theoretical task load W_j^{end} that would be attained if task T_j were to be processed next by operator ℓ, under the same assumptions of the previous step, along with an additional assumption that ξ_ℓ is operator ℓ's current task load;
      if $W_j^{\text{end}} - \overline{W} > \beta$ **then**
        Set $\overline{\epsilon}_j = W_j^{\text{end}} - \overline{W} - \beta$ and $\beta = W_j^{\text{end}} - \overline{W}$
      else
        Set $\overline{\epsilon}_j = 0$
      if $-W_j^{\text{end}} + \underline{W} > \gamma$ **then**
        Set $\underline{\epsilon}_j = -W_j^{\text{end}} + \underline{W} - \gamma$ and $\gamma = -W_i^{\text{end}} + \underline{W}$
      else
        Set $\underline{\epsilon}_i = 0$
      if $t_j^{\text{end}} - T_H > \alpha$ **then**
        Set $\epsilon_j = t_j^{\text{end}} - T_H - \alpha$ and $\alpha = t_j^{\text{end}} - T_H$
      else
        Set $\epsilon_j = 0$.
      Calculate a score $\pi_j = (R_j - p_\alpha\epsilon_j - p_\beta\overline{\epsilon}_j - p_\gamma\underline{\epsilon}_j)\lambda^{t_j^{\text{end}}-\theta_j}$;
    Select $j^* \in \arg\max_j \pi_j$ and, if task T_{j^*} is not a null-task, assign task T_{j^*} to operator ℓ;
    Set $\theta_\ell = t_{j^*}^{\text{end}}$, $\xi_j = W_{j^*}^{\text{end}}$ and, if task T_{j^*} is not a null-task, remove task T_{j^*} from $\mathcal{T}$.
return Operator/Task Pairings

Since we have only used one scenario thus far, the Algorithm 5 suffers from the same

deficiency as traditional deterministic scheduling schemes in the presence of uncertainty. Namely, we have performed task assignments based on the assumption that the processing times for a single scenario are true. To incorporate some degree of robustness into the heuristic task assignment scheme, we propose Algorithm 6 (see Figure 3.6).

This new algorithm has the added feature that it bases the final target assignment on a series of heuristic assignment operations, and assigns to each task the agent that is most likely to comprise the best task-operator assignment. One other small detail that must be addressed is the procedure to be used if there is a "tie" between two or more operators when executing the final step of Algorithm 6[1]. One possibility is to randomly select one of the operators as the final assignment. Although valid, this strategy does not take into account tasks that are already assigned to the set of operators that are tied. A potentially better approach in the case of ties is to assign the task according to either the global end times of the task, or the number of tasks that are already in the various operator's assignment. For example, if there is a tie between operators 1 and 2, but operator 1 has already been assigned 5 tasks and operator 2 has not been assigned any task, then it may be beneficial to give the task to operator 2. Thorough comparisons between these variations is left as a topic of future research.

Algorithm 6: *Multi-Scenario Task Assignment*

Input : $\mathcal{T} \coloneqq \{T_j\}_{j\in\{1,\dots,M+1\}}$; $\{W_{0,\ell}\}_{\ell\in\{1,\dots,L\}}$; Q, p_α, p_β, p_γ
Output : Operator/Task Pairings

1 Generate a set of Q scenarios, and initialize a matrix `counts` as a $M \times L$ matrix of zeros;
for *Each scenario q* **do**
2 Call Algorithm 6 to obtain a target-task assignment;
for *Each operator ℓ and each task T_j assigned to operator ℓ* **do**
3 Increment the (j,ℓ)-th element of the matrix `counts` by 1
4 Obtain final pairings: for each $j \in \{1,\dots,M\}$, assign T_j to agent $\ell^* \in \arg\max_\ell$ `counts`$(j,\cdot)$.
return Operator/Task Pairings

[1] "Ties" can also occur, in theory, in step 12 of Algorithm 5, though it is highly unlikely in practice. If such a scenario arises, a similar tie-breaking procedure should be used

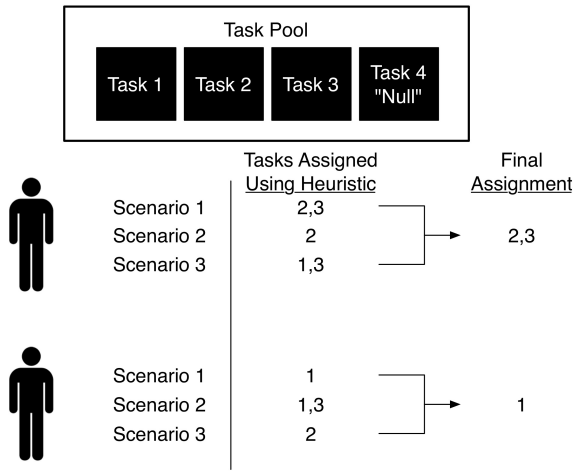

Figure 3.6: Schematic of the proposed task assignment process for a sample mission. An heuristic assignment step is run for each available scenario. Then, to obtain final operator/task pairings, each task is paired with the operator to which it was most often assigned while looping through the individual scenarios.

Figure 3.7 shows a performance comparison between 3 solution methods for a sample 3-operator scheduling problem, using a "least number of tasks" tie-breaking procedure. The solution methods used are (i) the "naive" approach of Section 3.1.4, (ii) the full heuristic assignment and solution approach of the present section, and (iii) a "reward-only" heuristic assignment strategy that is analogous to the full heuristic strategy, except scores are assigned to tasks solely based on the rewards R_i during assignment, i.e., the strategy is the same as that discussed above, except the value of $\underline{\epsilon_i}$, $\overline{\epsilon_i}$, and ϵ_i are always taken to be 0. The plots illustrate the average over 10 simulation runs of the ratio between the computation times, obtained nominal rewards, and the task load violations for the heuristic strategies to the respective variables for the naive strategy. Notice that the computation times for the heuristic strategies is only a small fraction of that necessary for the naive strategy, while the performance remains mostly unchanged. In fact, the heuristic strategies even out-performed the naive strategy in some cases. The reason that this is possible is again due to the nature of the heuristic solvers used. To prevent excessively long computation times, a bound was placed on number of nodes

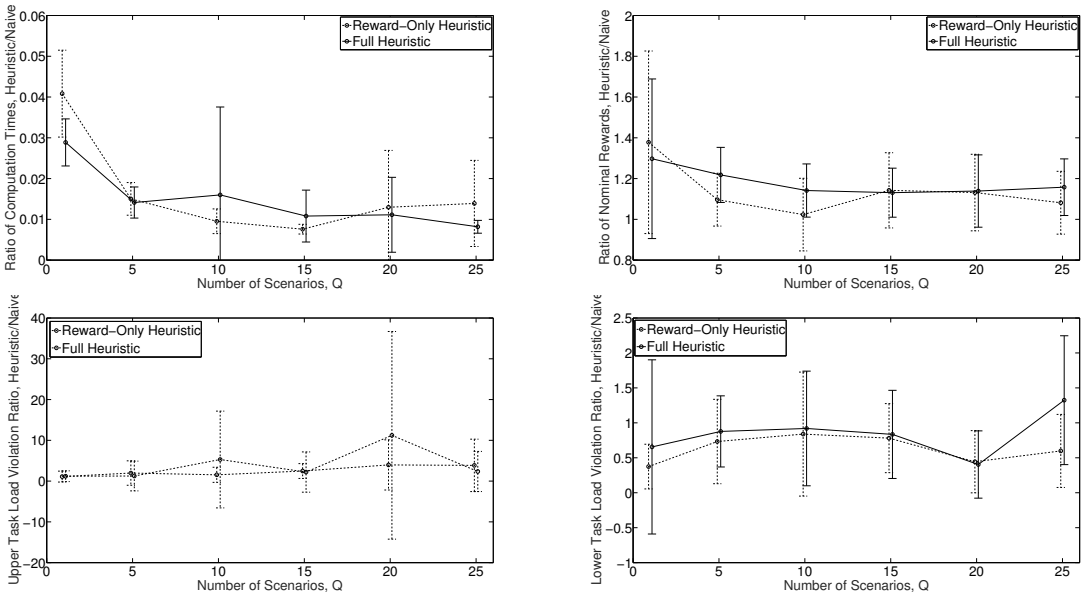

Figure 3.7: Performance of heuristic task assignment methods (Algorithm 6) in comparison to "naive" scheduling which solves the full, joint optimization problem (3.28).

that `intlinprog` solver was permitted to explore during its implicit branch and bound solution phase (10^5). This bound sometimes limited the quality of solutions produced by the naive strategy, allowing it to be out-performed by the heuristic method. This result only furthers the case that heuristic methods may be preferable to the naive approach.

Remark 8 (Task Assignment) *Algorithm 6 assigns tasks to operators on the basis of which operator was assigned a given task most often upon iterating through the various scenarios. Loosely, this is a sort of "maximum likelihood" approach in the sense that it selects, for each task, the operator that is most likely to be assigned to that task, given the realized samples of the processing time distributions.*

3.4 Chapter Summary

In the context of human supervisory missions involving sequential task analysis, it is crucial to develop task scheduling methodologies that (i) take human cognitive require-

ments into account, (ii) are robust to uncertainty in system and modeling parameters, (iii) can incorporate a wide range of design goals, and (iv) are simple enough to practically implement. Our presented MILP framework can potentially accomplish all of these goals. Indeed, we have illustrated how this framework can incorporate task load, introduce robustness, and expand to handle a number of additional layers of complexity, while remaining within a straightforward, well-studied, familiar theoretical framework.

With this preliminary work in hand, future work should focus on verifying the utility of these scheduling frameworks through human subjects experiments. In particular, these studies should seek to (i) verify that performance actually does, in fact, improve with the introduction of such scheduling systems, (ii) investigate whether simple task load models are sufficient for these applications or if more elaborate models should be considered, and (iii) identify any other unforeseen practical or performance issues. With this knowledge, additional theoretical methods can be explored on how to incorporate additional complexities and tailor system designs to particular missions. Furthermore, we envision that this kind of technology could be applied to a multitude of other domains, including healthcare and manufacturing applications. In addition to the suggestions throughout the text, other interesting future research avenues include a thorough investigation of how to choose optimal parameters for a given application domain or set of design goals, a study of the domain of applicability of the various solution methods, e.g., a study of the conditions under which estimation is useful, and a comparison between different heuristics for task assignment, given the time scales on which a particular mission operates.

Part II

Sensor-Focused Methods

Chapter 4

Cloud-Supported Coverage Control for Multi-Agent Surveillance

A large component of effective supervisory control systems, particularly those involving mobile sensors, is an intelligent coordination scheme to govern the behavior of autonomous agents. Recall that human operators in supervisory systems do not govern the low-level behavior of their robotic partners, but instead provide only periodic input in the form of high-level coordination commands. As such, mobile sensors that are present in the system must be able to act, in large part, as independent entities, choosing actions that support high-level mission goals without requiring constant operator interaction.

This chapter, along with the following chapter, shifts the focus momentarily away from the human operator, and focuses instead on the design of high-level coordination schemes for teams of mobile sensors. In essence, the schemes discussed in these chapters seek to improve overall system performance by enhancing the efficiency and effectiveness by which the automated agents (namely, mobile sensors) operate. In general, the appropriate control scheme will be dependent upon the particular task at hand, and thus will depend on the mission specifications, the operational environment, available hardware, among

other factors. The discussions of this dissertation are primarily motivated by supervisory surveillance systems involving teams of UAVs; thus, we focus on representative sub-problems that commonly arise in this domain.

We start our discussion of sensor-focused methods by developing a coordination scheme for a mission in which a team of mobile agents (e.g.,UAVs) must continuously monitor a planar, non-discrete region of interest to look for targets or monitor some event. We focus on a *decomposition-based* approach, where each agent is dynamically assigned surveillance responsibility of a particular region. Since peer-to-peer communication is often difficult or impossible in realistic supervisory systems (see Section 1.1), we develop a *cloud*-based approach to coverage region assignment, in which agents are only required to sporadically communicate with a central repository or *cloud* in an unplanned fashion (in supervisory systems, the cloud typically is a server that is housed at the same location as the human operator, e.g., a helicopter or ground control station). This is in contrast to typical load-balancing strategies that require complete or pairwise updates.

4.1 Mission Overview and Solution Approach

A team of N mobile agents[1], each equipped with an on-board sensor, is tasked with persistently monitoring a non-trivial, planar region of interest. The primary goal of the mission is to collect sensor data about some dynamic event or characteristic, e.g., the presence of an intruder. Collected data is periodically uploaded to the cloud. Agents must move within the environment to obtain complete sensory information. Ideally, agent motion should be coordinated so that (i) the sensing workload is balanced across agents, (ii) no subregion goes unobserved indefinitely, (iii) agents never collide (have sensor overlap), and (iv) the search is biased toward regions of greater interest. To

[1]Each agent is uniquely paired with a coverage region, so the quantity N represents both the number of agents and the number of regions in subsequent partitioning operations

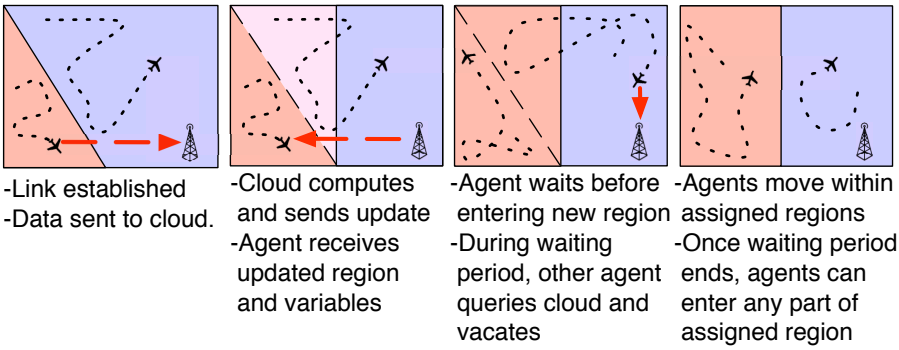

Figure 4.1: Illustration of the proposed decomposition-based, cloud-supported coverage control and surveillance strategy. There are two primary components to the framework: The partitioning component (executed on the cloud) manages coverage regions and introduces logic to prevent collisions, while the trajectory planning component (executed onboard each agent) governs agent motion.

achieve these goals, we propose a decomposition-based approach in which each agent's motion is restricted to lie within a dynamically assigned *coverage region*. The partitioning component (operating on the cloud), defines these coverage regions and provides high-level restrictions on agent motion through the manipulation of surveillance parameters, while the trajectory planning component (operating on-board each agent) incrementally constructs agent motion. We assume only asynchronous, cloud-agent data transfer, i.e., agents sporadically exchange data with the cloud, in which inter-exchange times are not specified *a priori*, but are subject to an upper bound.

Broadly, our framework operates as follows (Figure 4.1). Initial coverage variables are communicated to the agents prior to deployment, i.e., initial information is known to each agent at the mission onset. Once in the field, agents communicate sporadically with the cloud. During each agent-cloud exchange, the cloud calculates a new coverage region solely for the communicating agent, along with a set of timing and surveillance parameters that serve to govern the agent's high-level motion behavior, and transmits the update. The update algorithm also alters relevant global variables maintained by the cloud. Upon update completion, the data-link is terminated and the agent follows a trajectory found

via its onboard planner. Notice that this structure is a type of *event-triggered* control, since high-level updates only occur in the event of an agent-cloud exchange.

4.2 Problem Setup

The cloud, as well as each agent, has its own local processor. "Global" information is stored on the cloud, while each agent only stores information pertinent to itself.

Convention 1 *In this chapter, the subscripts i, j, or ℓ denote an entity or set element relevant to agent (i.e., sensor or UAV) i, j, or ℓ, resp. The superscript 'A' denotes an entity that is stored by the agent's local, i.e. on-board, processor.*

A storage summary is shown in Table 4.1. We expand on these, and define other relevant mathematical constructs here.

4.2.1 Mathematical Constructs and Definitions

Agent Dynamics: Each agent (sensor) i is a point mass that moves with maximum possible speed $s_i > 0$. Define $\mathbf{s} \coloneqq \{s_i\}_{i=1}^N$.

Communication Protocol: Each agent periodically exchanges data with the cloud. Assume the following:

1. each agent can identify itself to the cloud, and transmit/receive data,
2. there is a lower bound $\underline{\Delta} > 0$ on the time between any two successive exchanges involving the cloud, and
3. there is an upper bound $\overline{\Delta} > 0$ on the time between any *single agent's* successive exchanges with the cloud.

Table 4.1: Storage Summary for the Cloud-Supported Architecture

Stored on the Cloud	
Variable	Description
$G \coloneqq (V, E)$	Graphical representation of the environment
$\boldsymbol{P}$	N-covering of V $(\boldsymbol{P} \in \mathrm{Cov}_N(V))$
$\boldsymbol{c}$	Set of generators $\left(\boldsymbol{c} \in V^N\right)$
$\mathbf{ID}$	Set of identifiers $\left(\mathbf{ID} \in \{1, \ldots, N\}^{\lvert V \rvert}\right)$
$\boldsymbol{\Gamma}$	Set of timers $\left(\boldsymbol{\Gamma} \in \mathbb{R}^N_{\geq 0}\right)$
$\boldsymbol{\omega}$	Set of most recent exchange times $\left(\boldsymbol{\omega} \in \mathbb{R}^N_{\geq 0}\right)$
Φ	Global likelihood $(\Phi : V \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0})$

Stored by Agent i	
Variable	Description
$G \coloneqq (V, E)$	Graphical representation of the environment
P_i^A	Coverage region $\left(P_i^A \subset V\right)$
c_i^A	Coverage region generator $\left(c_i^A \in V\right)$
$P_i^{A,\mathrm{pd}}$	Set of "recently added" vertices $\left(P_i^{A,\mathrm{pd}} \subseteq P_i^A\right)$
γ_i^A	Local timing parameter $\left(\gamma_i^A \in \mathbb{R}\right)$
ω_i^A	Most recent exchange time $\left(\omega_i^A \in \mathbb{R}_{\geq 0}\right)$
Φ_i^A	Local likelihood $\left(\Phi_i^A : V \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}\right)$

Assume that agent-cloud exchanges occur instantaneously, and notice that condition 2 implies no two exchanges (involving any agents) occur simultaneously[2]. Since computation time is typically small in comparison to inter-exchange times and exchanges occur sporadically, these assumptions are without significant loss of generality.

Environment: Consider a bounded surveillance environment as a finite grid of disjoint, non-empty, simply-connected subregions. We represent the grid as a weighted graph $G \coloneqq (V, E)$, where V is the set of vertices (each representative of a unique grid element), and E is the edge set comprised of undirected, weighted edges $\{k_1, k_2\}$ spanning

[2]Mathematically, the bound $\underline{\Delta}$ also prevents *zeno behavior*

vertices representing *adjacent*[3] grid elements. Let the weight associated to $\{k_1, k_2\}$ be some finite upper bound on the minimum travel distance between any point in the grid element associated to k_1 to any point in the grid element associated to k_2 along a path that does not leave the union of the two elements. Locations of environmental obstacles and prohibited areas are known and are not included in the graphical representation G.

Consider $V' \subseteq V$. A vertex $k_1 \in V$ is *adjacent* to V' if $k_1 \notin V'$ and there exists $\{k_1, k_2\} \in E$ with $k_2 \in V'$. Define $G(V') \coloneqq (V', E')$ as the subgraph of G induced by the vertex set V'. A *path* on $G(V')$ between $k_1, k_n \in V'$ is a sequence $(k_1, k_2, \ldots, k_n)$, where $k_1, k_2, \ldots, k_n \in V'$ and $\{k_r, k_{r+1}\} \in E'$ for $r \in \{1, \ldots, n-1\}$. We say V' is *connected* if a path exists in $G(V')$ between any $k_1, k_2 \in V'$. Let $d_{V'} : V' \times V' \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ be the standard graph distance on $G(V')$, i.e., the length of a shortest weighted path in $G(V')$ (if none exists, $d_{V'}$ takes value ∞). Notice that $d_V(k_1, k_2) \leq d_{V'}(k_1, k_2)$ for any $k_1, k_2 \in V'$. Also let $d_{V'}$ denote the map $d_{V'} : V' \times 2^{V'} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, where $d_{V'}(k, V'')$ is the length of a shortest weighted path in $G(V')$ between k and any vertex in V''.

Coverage Regions: An *N-covering* of V is a family $\boldsymbol{P} \coloneqq \{P_i \subseteq V\}_{i=1}^N$ satisfying (i) $\bigcup_{i=1}^N P_i = V$, and (ii) $P_i \neq \emptyset$ for all i. Define $\mathrm{Cov}_N(V)$ as the set of all possible N-coverings of V. An *N-partition* of V is an N-covering that also satisfies (iii) $P_i \bigcap P_j = \emptyset$ for all $i \neq j$. An N-covering or N-partition $\boldsymbol{P}$ is *connected* if each P_i is connected. In what follows, the cloud maintains an N-covering $\boldsymbol{P}$ of V, and surveillance responsibilities are assigned by pairing each agent i with $P_i \in \boldsymbol{P}$ (called agent i's *coverage region*). Each agent maintains a copy P_i^A of P_i. The cloud also stores a set $\boldsymbol{c} \coloneqq \{c_i \in V\}_{i=1}^N$ (c_i is the *generator* of P_i), and each agent i maintains a copy c_i^A of c_i.

Identifiers, Timers, and Auxiliary Variables: The proposed algorithms also require

[3]Travel between the elements without leaving their union is possible

some logic and timing variables. To each $k \in V$, assign an *identifier* $\mathrm{ID}_k \in \{1, \ldots, N\}$. Define $\mathbf{ID} \coloneqq \{\mathrm{ID}_k\}_{k \in V}$, and let $\boldsymbol{P}^{\mathrm{ID}} \coloneqq \{P_i^{\mathrm{ID}}\}_{i=1}^N$, where $P_i^{\mathrm{ID}} \coloneqq \{k \in V \mid \mathrm{ID}_k = i\}$. Notice $\boldsymbol{P}^{\mathrm{ID}}$ is an N-partition of V. For each agent i, define a timer Γ_i having dynamics $\dot{\Gamma}_i = -1$ if $\Gamma_i \neq 0$, and $\dot{\Gamma}_i = 0$ otherwise. Define $\boldsymbol{\Gamma} \coloneqq \{\Gamma_i\}_{i=1}^N$. Each agent i maintains a local timing variable γ_i^A. Even though γ_i^A plays a similar role to Γ_i, note that γ_i^A is constant unless explicitly updated, while Γ_i has autonomous dynamics. Next, the cloud maintains a set $\boldsymbol{\omega} \coloneqq \{\omega_i\}_{i=1}^N$, where ω_i is the time of agent i's most recent exchange with the cloud. Each agent maintains a copy ω_i^A of ω_i. Finally, each agent stores a subset $P_i^{A,\mathrm{pd}} \subseteq P_i^A$ which collects vertices that have recently been added to P_i^A.

Likelihood Functions: The likelihood of a relevant event occurring within any subset of the surveillance region V is maintained on the cloud in the form of a time-varying probability mass function[4] $\Phi : V \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$. For simplicity, assume that, at any t, the instantaneous support, $\mathtt{supp}(\Phi(\cdot, t))$, equals V.

Define each agent's *local likelihood* $\Phi_i^A : V \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as the function that, loosely, represents the agent's local belief regarding events. Specifically, define

$$\Phi_i^A(k,t) = \begin{cases} \Phi(k,t), & \text{if } k \in P_i^A \text{ and} \\ & \left(t - \omega_i^A \geq \gamma_i^A \text{ or } k \notin P_i^{A,\mathrm{pd}}\right), \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

The conditions defining Φ_i^A are understood as follows: at some time t, an element $k \in V$ only belongs to $\mathtt{supp}(\Phi_i^A(\cdot, t))$ if (i) $k \in P_i^A$, and (ii) sufficient time has passed since k was first added to P_i^A, as determined by the parameters γ_i^A, ω_i^A, and $P_i^{A,\mathrm{pd}}$. Notice that,

[4] $\Phi(\cdot, t)$ is a probability mass function for any time t.

in general, each Φ_i^A will be different[5] from Φ.

Remark 9 (Global Data) *If global knowledge of Φ is not available instantaneously to agent i, Φ_i^A can alternatively be defined by replacing $\Phi(k,t)$ in (4.1) by $\Phi(k,\omega_i^A)$. All subsequent theorems hold under this alternative definition.*

Remark 10 (Data Storage) *The cost of storing a graph as an adjacency list is $O(|V|+|E|)$. The generator set $\boldsymbol{c}$, each element of $\boldsymbol{P}$, and the identifier set $\boldsymbol{ID}$ can be stored as integral vectors. The timer set $\boldsymbol{\Gamma}$ and the set $\boldsymbol{\omega}$ are are stored as real vectors, while Φ is stored as a time-varying real vector. Thus, the cost of storage on the cloud is $O(N|V|+|E|)$. Similarly, each agent's local storage cost is $O(|V|+|E|)$.*

4.3 Dynamic Coverage Update Scheme

We adopt following convention for the remainder of this chapter.

Convention 2 *Suppose that:*

1. $\min\emptyset \coloneqq \max\emptyset \coloneqq 0$, *and*
2. *given a specific time instant, superscripts '-' or '+' refer to the value of the associated variable before or after the instant in question, resp.*

4.3.1 Additive Set

We start with a definition.

Definition 1 (Additive Set) *Given $k \in P_i^{ID}$, the* additive set $P_i^{\text{add}}(k) \subseteq V$ *is the largest connected subset satisfying:*

[5] Φ_i^A need not be normalized and thus may not be a time-varying probability mass function in a strict mathematical sense

1. $P_i^{ID} \subseteq P_i^{\text{add}}(k)$, *and*

2. *for any* $h \in P_i^{\text{add}}(k) \cap P_j$, *where* $j \neq i$*:*

   (a) $\Gamma_j = 0$, *and*

   (b) $\frac{1}{s_i} d_{P_i^{\text{add}}(k)}(h, k) < \frac{1}{s_j} d_{P_j}(h, c_j)$.

The following characterizes well-posedness of Definition 1.

Proposition 1 (Well-Posedness) *If* P_i^{ID} *is connected and disjoint from* $\bigcup_{j \neq i} P_j$*, then* $P_i^{\text{add}}(k)$ *exists and is unique for any* $k \in P_i^{ID}$.

Proof: Under the specified conditions, P_i^{ID} is connected and satisfies conditions 1 and 2 in Definition 1; $P_i^{\text{add}}(k)$ is the unique, maximally connected superset of P_i^{ID} satisfying the same conditions. ■

It is important to note that, under the conditions of Proposition 1, if $h \in P_i^{\text{add}}(k)$, then (i) $\max\{\Gamma_j \,|\, j \neq i, h \in P_j\} = 0$, and (ii) there is a path from k to h in $G(P_i^{\text{add}}(k))$ that is shorter than the optimal path spanning c_j and h within $G(P_j)$, for any $j \neq i$ with $h \in P_j$.

4.3.2 Cloud-Supported Coverage Update

We start by defining a cost/performance function that we can use to evaluate a given N-covering $\boldsymbol{P}$. Define the cost function $\mathcal{H} : V^N \times \text{Cov}_N(V) \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ by

$$\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t) = \sum_{k \in V} \min_i \left\{ \frac{1}{s_i} d_{P_i}(k, c_i) \,|\, k \in P_i \right\} \Phi(k, t).$$

The intuition behind the definition of $\mathcal{H}$ is as follows: If (i) each agent is solely responsible for events within its own coverage region, and (ii) events occur proportionally to Φ, then $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t)$ is the expected time required for an agent to reach a randomly occurring event from its region generator at time t; related functions are studied in [1, 104, 108].

Algorithm 7: Cloud-Supported Coverage Update

Data: t_0, $\boldsymbol{P}$, $\boldsymbol{c}$, Φ, $\boldsymbol{\omega}$, $\overline{\Delta}$, Δ_{H}, $\boldsymbol{\Gamma}$, **ID**, $\boldsymbol{s}$
Result: $\boldsymbol{P}$, $\boldsymbol{c}$, P_i^A, c_i^A, $P_i^{A,\mathrm{pd}}$, $\boldsymbol{\Gamma}$, γ_i^A, Φ_i^A, $\boldsymbol{\omega}$, ω_i^A, **ID**
begin
% Initialize, remove regions others have claimed
Initialize $\boldsymbol{P}^* = \boldsymbol{P}^{\mathrm{test}} = \boldsymbol{P}$, $\boldsymbol{c}^* = \boldsymbol{c}^{\mathrm{test}} = \boldsymbol{c}$
Set $P_i^* = P_i^{\mathrm{test}} = P_i^{\mathrm{ID}}$
% If timer is non-zero and no regions have been claimed since last update, perform trivial update
if $\Gamma_i > 0$ *and* $P_i^* = P_i$ **then**
  Set $\gamma_i^A = \gamma_i^A - t_0 + \omega_i$ and $\omega_i^A = \omega_i = t_0$
else
  % Iterate through generator locations to find cost-minimizing configuration
  for $k \in P_i^{ID}$ **do**
    Set $P_i^{\mathrm{test}} = P_i^{\mathrm{add}}(k)$ and $c_i^{\mathrm{test}} = k$
    if $\mathcal{H}(\boldsymbol{c}^{\mathrm{test}}, \boldsymbol{P}^{\mathrm{test}}, t_0) < \mathcal{H}(\boldsymbol{c}^*, \boldsymbol{P}^*, t_0)$ **then**
      Set $\boldsymbol{P}^* = \boldsymbol{P}^{\mathrm{test}}$, $\boldsymbol{c}^* = \boldsymbol{c}^{\mathrm{test}}$
  % Update timers and variables
  Set $P_i^{A,\mathrm{pd}} = P_i^* \backslash P_i^{\mathrm{ID}}$
  Call Alg. 8 and obtain output $\Phi_i^A, \omega, T, \gamma_i^A$
  Set $P_i = P_i^A = P_i^*$, $c_i = c_i^A = c_i^*$, $\omega_i^A = \omega_i$
  for $k \in P_i$ **do** Set $\mathrm{ID}_k = i$
return $\boldsymbol{P}$, $\boldsymbol{c}$, P_i^A, c_i^A, $P_i^{A,\mathrm{pd}}$, $\boldsymbol{\Gamma}$, γ_i^A, Φ_i^A, $\boldsymbol{\omega}$, ω_i^A, **ID**

We are now in a position to define the main algorithm of this chapter: Algorithm 7 defines the operations performed on the cloud when agent i makes contact at time t_0. In the algorithm, the input $\Delta_{\mathrm{H}} > 0$ is a constant parameter[6], and the auxiliary variables are made up of components relevant to each agent: $\boldsymbol{P}^* \coloneqq \{P_i^*\}_{i=1}^N$, $\boldsymbol{c}^* \coloneqq \{c_i^*\}_{i=1}^N$, $\boldsymbol{P}^{\mathrm{test}} \coloneqq \{P_i^{\mathrm{test}}\}_{i=1}^N$, and $\boldsymbol{c}^{\mathrm{test}} \coloneqq \{c_i^{\mathrm{test}}\}_{i=1}^N$.

[6] Δ_H represents, loosely, the amount of time an agent must hold a vertex before it can be re-assigned. More precise characterization is contained later in Section 4.3.4.

Algorithm 8: Timer Update

Data: t_0, $\boldsymbol{P}$, $\boldsymbol{P}^*$, $\boldsymbol{c}^*$, $P_i^{A,\mathrm{pd}}$, Φ, $\boldsymbol{\omega}$, $\overline{\Delta}$, Δ_{H}, $\boldsymbol{\Gamma}$, $\boldsymbol{s}$
Result: $\Phi_i^A, \boldsymbol{\omega}, \boldsymbol{\Gamma}, \gamma_i^A$
begin

% Find max time to return to coverage region

$\Delta_i^{\mathrm{Bf}} := \max\left\{\frac{1}{s_i} d_{P_i}(k, P_i^* \backslash P_i^{A,\mathrm{pd}}) \,|\, k \in P_i \backslash P_i^*\right\}$

% Find max time for agents to vacate acquired regions, redefine timers to ensure communication

for *Each* $j \neq i$ *satisfying* $P_j \cap P_i^* \neq \emptyset$ **do**
  $\Delta_j^{\mathrm{Bf}} := \max\left\{\frac{1}{s_j} d_{P_j}(k, P_j \backslash P_i^*) \,|\, k \in P_j \cap P_i^*\right\}$
  Set $\Gamma_j = \omega_j + \overline{\Delta} - t_0$

% Select maximum and redefine variables

Find $\Delta_{\max}^{\mathrm{Bf}} = \max_{j \neq i, P_j \cap P_i^* \neq \emptyset} \{\omega_j + \overline{\Delta} + \Delta_j^{\mathrm{Bf}} - t_0\}$
Redefine $\Delta_{\max}^{\mathrm{Bf}} = \max\{\Delta_{\max}^{\mathrm{Bf}}, \Delta_i^{\mathrm{Bf}}\}$
Set $\Gamma_i = \Delta_{\max}^{\mathrm{Bf}} + \Delta_{\mathrm{H}}$, $\gamma_i^A = \Delta_{\max}^{\mathrm{Bf}}$, $\omega_i = t_0$
Construct Φ_i^A (Eq. (4.1)) with updated variables
return $\Phi_i^A, \boldsymbol{\omega}, \boldsymbol{\Gamma}, \gamma_i^A$

4.3.3 Well-Posedness

Consider the following initialization assumption.

Assumption 1 (Initialization) *The following properties are satisfied when $t = 0$:*

1. $\boldsymbol{P}$ *is a connected N-partition of V,*
2. $\boldsymbol{P} = \boldsymbol{P}^{ID}$, *and*
3. *for all $i \in \{1, \ldots, N\}$,*
   (a) $P_i = P_i^{\mathrm{A}}$,
   (b) $c_i = c_i^A \in P_i^A$,
   (c) $P_i^{A,\mathrm{pd}} = \emptyset$,
   (d) $\Gamma_i = \omega_i = \omega_i^A = 0$, *and*
   (e) $\gamma_i^A = -\Delta_{\mathrm{H}}$.

Notice that conditions 1 and 3b of Assumption 1 together imply that $c_i \neq c_j$ for any $j \neq i$. The following theorem, whose proof is postponed until Appendix A, characterizes the well-posedness of Algorithm 7.

Theorem 1 (Well-Posedness) *Under Assumption 1, a scheme in which, during each agent-cloud exchange, the cloud executes Algorithm 7 to update relevant global and local variables is well-posed. That is, any operations required by Algorithm 7 are always well-posed at the time of execution.*

Algorithm 7 does not ensure that coverage regions (elements of $\boldsymbol{P}$) remain disjoint. It does, however, guarantee that the N-covering $\boldsymbol{P}$, the local coverage regions $\boldsymbol{P}^A := \{P_i^A\}_{i=1}^N$, and the local likelihoods $\{\Phi_i^A\}_{i=1}^N$ retain properties that are consistent with a

decomposition-based scheme. Namely, the coverings $\boldsymbol{P}$ and $\boldsymbol{P}^A$ maintain connectivity, and each Φ_i^A has support that is disjoint from that of all other local likelihoods, yet evolves to provide reasonable global coverage. Further, Algorithm 8 ensures that agents can "safely" vacate areas that are re-assigned before newly assigned agents enter, i.e., they can vacate areas that are re-assigned without introducing a collision or redundant sensing risk. We expand upon these ideas in the following two subsections. For clarity, proofs of all results in these two sections are postponed until Appendix A

4.3.4 Set Properties

The next result formalizes key set properties induced by Algorithm 7.

Theorem 2 (Set Properties) *Suppose Assumption 1 holds, and that, during each agent-cloud exchange, the cloud executes Algorithm 7 to update relevant global and local variables. Then, the following hold at any time $t \geq 0$:*

1. $\boldsymbol{P^{ID}}$ *is a connected N-partition of V,*
2. $\boldsymbol{P}$ *is a connected N-covering of V,*
3. $c_i \in P_i$ *and* $c_i \neq c_j$ *for any* $i \neq j$,
4. $\boldsymbol{supp}(\Phi_i^A(\cdot, t)) \subseteq P_i$ *for any* i, *and*
5. $\bigcap_{i=1}^N \boldsymbol{supp}(\Phi_i^A(\cdot, t)) = \emptyset$

When the cloud makes additions to an agent's coverage region, newly added vertices are not immediately included in the instantaneous support of the agent's local likelihood. If agent movement is restricted to lie within this support, the aforementioned delay temporarily prohibits the agent from exploring newly added regions, allowing time for

others to vacate. Conversely, when regions are removed from an agent's coverage region, Algorithm 7 ensures a "safe" path, i.e., a path with no collision risk, exists and persists long enough for the agent to vacate. Let $\overline{d} \coloneqq \max_i \frac{1}{s_i} \sum_{\{k_1,k_2\}\in E} d_V(k_1,k_2)$, and define agent i's *prohibited region* at time t as $\texttt{Proh}_i(t) \coloneqq \{k \in V \,|\, k \notin \texttt{supp}(\Phi_i^A(\cdot,t))\}$. With this terminology, we formalize the previous discussion here.

Theorem 3 (Coverage Quality) *Suppose Assumption 1 holds, and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 7. Then, for any $k \in V$ and any $t \geq 0$:*

1. *k belongs to at least one agent's coverage region P_i,*
2. *if $k \in \boldsymbol{\texttt{Proh}}_i(t) \cap P_i$ for some i, then there exists t_0 satisfying $t < t_0 < t + \overline{\Delta} + \overline{d}$ such that, for all $\bar{t} \in [t_0, t_0 + \Delta_{\mathrm{H}}]$, the vertex k belongs to the set $P_i \backslash \boldsymbol{\texttt{Proh}}_i(\bar{t})$, and*
3. *if k is removed from P_i at time t, then, for all $\bar{t} \in \left(t, t + \frac{1}{s_i} d_{P_i^-}\left(k, P_i^{ID,-}\right)\right]$, we have*
   (a) *$P_i^{ID,-} \subseteq P_i$, and*
   (b) *there exists a length-minimizing path on $G(P_i^-)$ from k into $P_i^{ID,-}$, and all of the vertices along any such path (except the terminal vertex) belong to the set $\boldsymbol{\texttt{Proh}}_{ID_k^+}(\bar{t}) \backslash \bigcup_{j \neq ID_k^+} P_j$.*

Theorems 2 and 3 allow Algorithm 7 to operate within a decomposition-based framework to provide reasonable coverage with inherent collision avoidance. Indeed, when agents avoid prohibited regions, the theorems imply that each agent (i) can visit its entire coverage region (connectedness), (ii) allows adequate time for other agents to vacate newly assigned regions, and (iii) has a "safe" route into the remaining coverage region if its current location is removed during an update.

Remark 11 (Coverage Variables) *If Assumption 1 holds and updates are performed with Algorithm 7, then $P_i = P_i^A$ and $c_i = c_i^A$ for all i and all t. Thus, both Theorem 2 and Theorem 3 are equivalently stated by replacing P_i with P_i^A and c_i with c_i^A in their respective theorem statements.*

Remark 12 (Bounds) *Theorem 3 holds if $\overline{d}$ is replaced by any upper bound on the distance between any possible pair of vertices within any possible connected subgraph.*

4.3.5 Convergence Properties

Our proposed strategy differs from [1] due to logic, i.e., timing parameters, etc., that ensures effective operation within a decomposition-based framework. Note also that $\mathcal{H}$ differs from previous partitioning cost functions in [1, 104, 108], since it uses subgraph, rather than global graph, distances. As such, convergence properties of the algorithms herein do not follow readily from existing results. As such, we explore the convergence properties of our algorithms in detail here. Consider the following definition.

Definition 2 (Pareto Optimality) *The pair $(\boldsymbol{c}, \boldsymbol{P})$ is Pareto optimal at time t if*

1. $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t) \leq \mathcal{H}(\bar{\boldsymbol{c}}, \boldsymbol{P}, t)$ *for any* $\bar{\boldsymbol{c}} \in V^N$, *and*
2. $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t) \leq \mathcal{H}(\boldsymbol{c}, \overline{\boldsymbol{P}}, t)$ *for any* $\overline{\boldsymbol{P}} \in Cov_N(V)$.

When Φ is time-invariant (and Assumption 1 holds), Algorithm 7 produces finite-time convergence of coverage regions and generators to a Pareto optimal pair. The limiting coverage regions are "optimal" in that they balance the sensing load in a way that directly considers the event likelihood. Further, the operation only requires sporadic and unplanned agent-cloud exchanges. We formalize this result here.

Theorem 4 (Convergence) *Suppose Assumption 1 holds and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 7. If Φ is time-invariant, i.e., $\Phi(\cdot, t_1) = \Phi(\cdot, t_2)$ for all t_1, t_2, then the N-covering $\boldsymbol{P}$ and the generators $\boldsymbol{c}$ converge in finite time to an N-partition $\boldsymbol{P}^*$ and a set $\boldsymbol{c}^*$, resp. The pair (c^*, P^*) is Pareto optimal at any time following convergence.*

We can further characterize convergence if we introduce the notion of (multiplicatively-weighted) Voronoi partitions and (generalized) centroid sets.

Definition 3 (Centroid Set) *Define the* centroid set *of a subset $V' \subseteq V$ at time $t \in \mathbb{R}_{\geq 0}$ as the set of elements in V' that minimize a* one-center function*, i.e.,*

$$C(V', t) := \arg\min_{h \in V'} \sum_{k \in V'} d_{V'}(k, h)\Phi(k, t).$$

Definition 4 (Multiplicatively-Weighted Voronoi Partition) *The N-covering $\boldsymbol{P}$ of V is a* multiplicatively weighted Voronoi partition *at time $t \in \mathbb{R}_{\geq 0}$, generated by $\boldsymbol{c}$ and weighted by $\boldsymbol{s}$, if $\boldsymbol{P}$ is an N-partition of Q and, for each i,*

1. *$c_i \in P_i$, and*
2. *$\frac{1}{s_i} d_V(k, c_i) \leq \frac{1}{s_j} d_V(k, c_j)$ for all $j \neq i$ and all $k \in P_i$.*

If the generators $\boldsymbol{c}$ also satisfy $c_i \in C(P_i, t)$ for all i, then $\boldsymbol{P}$ is also called centroidal*.*

The following proposition relates Pareto-optimal configurations to centroidal, (multiplicatively-weighted) Voronoi partitions.

Proposition 2 (Pareto Optimality and Voronoi partitions) *If a pair $(\boldsymbol{c}, \boldsymbol{P})$, where $\boldsymbol{c} \in V^N$ and $\boldsymbol{P} \in (2^V)^N$ is an N-partition of V, is Pareto optimal at time t by Defini-*

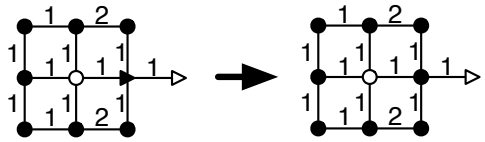

Figure 4.2: Assuming uniform density at the instant shown, the left diagram shows a centroidal Voronoi partition generated by the unfilled vertices (generators) and weighted uniformly, ie. $s_i = s_j$ for all $i, j \in \{1, \ldots, N\}$. Here, the shape of the vertices indicate which region the vertex belongs to, and the numbers represent edge weights. However, the left configuration is not Pareto optimal by Definition 2, as the cost $\mathcal{H}$ can be decreased by moving to the configuration on the right (fixing generators).

tion 2, then $\boldsymbol{P}$ *is also a centroidal, multiplicatively-weighted Voronoi partition generated by* $\boldsymbol{c}$ *and weighted by* $\boldsymbol{s}$*.*

Note that a general centroidal Voronoi partition $\boldsymbol{P}$ generated by $\boldsymbol{c}$ and weighted by $\boldsymbol{s}$ is not always Pareto optimal. Indeed, a counterexample is shown in Figure 4.2. Proposition 2 immediately leads to the following corollary to Theorem 4.

Corollary 1 (Limiting Configurations) *Suppose Assumption 1 holds and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 7. If* Φ *is time-invariant, i.e.,* $\Phi(\cdot, t_1) = \Phi(\cdot, t_2)$ *for all* t_1, t_2*, then the* N*-covering* $\boldsymbol{P}$ *and the generators* $\boldsymbol{c}$ *converge in finite time to an* N*-partition* $\boldsymbol{P}^*$ *and a set* $\boldsymbol{c}^*$*, resp. The* N*-covering* $\boldsymbol{P}^*$ *of* V *is a centroidal, multiplicatively-weighted Voronoi partition, generated by* $\boldsymbol{c}^*$ *and weighted by* $\boldsymbol{s}$*, at any time following convergence.*

4.4 Decomposition-Based Surveillance.

This section pairs the proposed partitioning framework with a generic, single-vehicle trajectory planner, forming the complete framework.

4.4.1 Complete Routing Algorithm

By Theorem 2, the support of each Φ_i^A (i) lies entirely within P_i^A, and (ii) is disjoint from the support of other local likelihoods. By Theorem 3, (i) any vertex only goes uncovered over bounded intervals, and (ii) the parameter Δ_{H} is a lower bound on the time that a recently uncovered vertex must remain covered before it can become uncovered again. These results suggest an intelligent routing scheme to achieve adequate coverage while maintaining collision avoidance that carefully restricts motion according to the instantaneous support of the local likelihood functions. This motivates the following assumption.

Assumption 2 (Agent Motion) *Each agent i has knowledge of its position at any time t, and its on-board trajectory planner operates under the following guidelines:*

1. *generated trajectories obey agent motion constraints,*
2. *trajectories are constructed incrementally and can be altered in real-time, and*
3. *the agent is never directed to enter regions associated with $\texttt{Proh}_i(t)$.*

Assume further that each agent precisely traverses generated trajectories.

Algorithm 9 presents the local (on-board) motion protocol for Agent i.

4.4.2 Collision Avoidance

Although Assumption 2 locally prevents agents from entering prohibited regions, dynamic coverage updates can still place an agent within its prohibited region when the vertex corresponding to its location is abruptly removed. If this happens, Algorithm 9 constructs a route from the agent's location back into a region where there is no collision risk. With mild assumptions, this construction (i) is well-defined, and (ii) does not

Algorithm 9: Motion Protocol for Agent i

Data: G, Φ_i^A, P_i^A, c_i^A, $P_i^{A,\mathrm{pd}}$, γ_i^A, ω_i^A
begin
  while *True* **do**
1 Increment trajectory via on-board planner
2 Follow trajectory
  if *Data link with the cloud* **then**
3 Set $P_i^{\mathrm{test}} = P_i^A$
4 Obtain updated variables from the cloud
  if *Location lies within* $P_i^{\mathrm{test}} \backslash P_i^A$ **then**
5 Find shortest path in the graph $G(P_i^{\mathrm{test}})$ from the currently occupied node into P_i^A
  while *Agent i is outside* P_i^A **do**
6 Follow the aforementioned route

present a transient collision risk. We formalize this result here (once again, we postpone proof until Appendix A).

Theorem 5 (Collision Avoidance) *Suppose Assumptions 1 and 2 hold, and that each agent's initial position lies within its initial coverage region P_i. If each agent's motion is locally governed according to Algorithm 9, where the update in line* 4 *is calculated on the cloud via Algorithm 7, then no two agents will ever collide.*

Remark 13 (Agent Dynamics) *We assume point mass dynamics for simplicity. However, all theorems herein also apply under alternative models, e.g., non-holonomic dynamics, provided that the environment is discretized so that (i) travel between adjacent grid elements is possible without leaving their union, (ii) agents can traverse the aforementioned paths at maximum speed, and (iii) edge weights accurately upper bound travel between adjacent regions. When these conditions are not met, Theorems 1, 2, 3, and 4 still apply, though Theorem 5 is no longer guaranteed.*

4.5 Numerical Examples

This section presents numerical examples to illustrate the proposed framework. In all examples, updates are performed on the cloud via Algorithm 7 during each agent-cloud exchange, and each agent's local processor runs the motion protocol in Algorithm 9. For incremental trajectory construction (Algorithm 9, line 1), all examples use a modified *Spectral Multiscale Coverage* (SMC) scheme [114], which creates trajectories to mimic ergodic dynamics while also locally constraining motion to lie outside of prohibited regions. Notice this planner satisfies Assumption 2. Initial region generators $\boldsymbol{c}$ were selected randomly (enforcing non-coincidence), and each agent was initially placed at its region generator. The initial covering $\boldsymbol{P}$ was created by calculating a weighted Voronoi partition, and remaining initial parameters were chosen to satisfy Assumption 1. Assume that relevant initial variables are uploaded to the agents' local servers prior to initial deployment, i.e., each agent has full knowledge of initial information at time 0. For each simulation, randomly chosen agents sporadically exchange data with the cloud. Agent-cloud exchange times were randomly chosen, with maximum inter-exchange time $\overline{\Delta}$.

4.5.1 Time-Invariant Likelihood

Consider a 4 agent mission, executed over a 100 x 100 surveillance region that is subject to a time-invariant, Gaussian likelihood centered near the bottom left corner. The region is divided into 400, 5 x 5 subregions. Regions are considered adjacent if they share a horizontal or vertical edge. Here, each agent had a maximum speed of 1 unit distance per unit time, and $\overline{\Delta} = 10$ time units. Figure 4.3 shows the evolution of the coverage regions for an example simulation run. Agent trajectories are shown by the colored lines. Note that Figure 4.3 only shows each agent i's *active* coverage region, i.e., $P_i \backslash \texttt{Proh}_i(t)$. The family of active coverage regions does not generally form an N-covering

of V; however, elements of this family are connected and never intersect as a result of inherent collision avoidance properties.

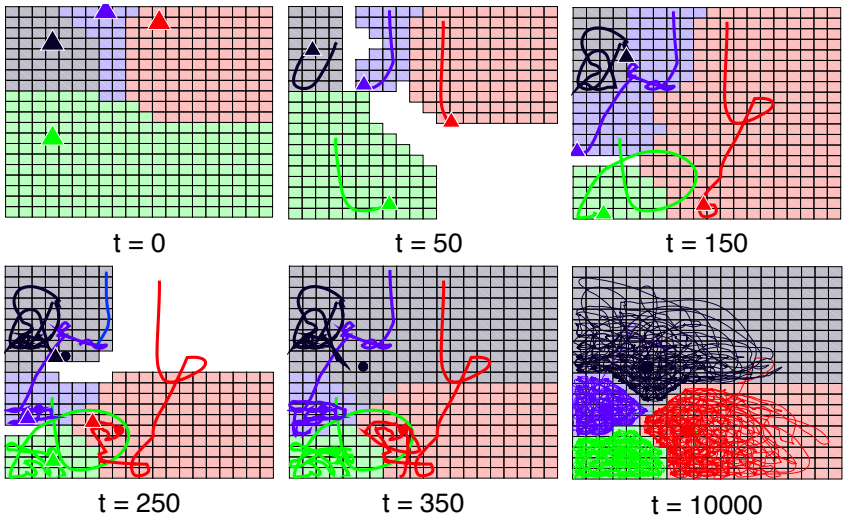

Figure 4.3: Illustration of a 4 agent example mission over a static Gaussian likelihood. Each agent's position, past trajectory, and active coverage region are shown with the colored triangle, line, and squares, resp.

The left plot in Figure 4.4 depicts the maximum time that any grid square went uncovered, i.e. did not belong to any agent's active covering, during each of 50 simulation runs. Here, the maximum amount of time that any region went uncovered was 186 units,

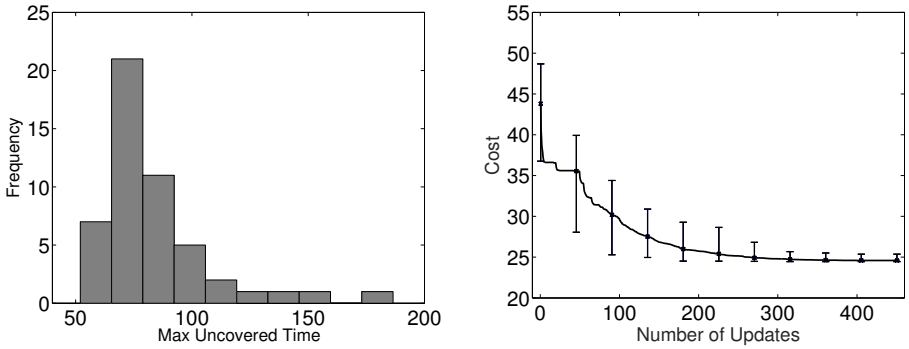

Figure 4.4: The maximum amount of time that any subregion went uncovered in each of 50 simulation runs (left), and the value of the cost $\mathcal{H}$ as a function of time, averaged over the same 50 runs (right), for the 4 agent sample mission.

though the maximum for most trials was less than 75 units. This is well-below the loose

bound $\overline{\Delta} + \overline{d} = 770$ predicted by Theorem 3 (see Remark 12). Note that this metric does not capture the time between the agent's actual visits to the grid-square, but rather the length of intervals on which no agent was *allowed* to visit the square. The time between visits is governed by the particular choice of trajectory planner and the parameter Δ_{H}.

The right plot in Figure 4.4 shows the mean values of the cost function $\mathcal{H}$, calculated over the same 50 simulations runs. Here, error bars represent the range of cost values at select points. The variance between runs is due to the stochastic nature of the agent-cloud exchange patterns. Notice the cost is non-increasing over time, eventually settling as the coverage regions/generators reach their limiting configuration, e.g., see Figure 4.3. These limiting configurations are each Pareto optimal and form a multiplicatively weighted Voronoi partition. The resultant coverage assignments provide load-balancing that takes into account the event likelihood. If the low-level trajectory planner biases trajectories according to the event likelihood, this results in desirable coverage properties. Under the modified SMC planner used here, the temporal distribution of agent locations closely resembles the spatial likelihood distribution in the limit, as shown in Figure 4.5.

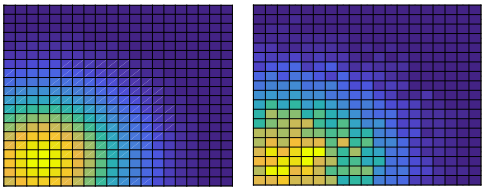

Figure 4.5: Comparison between the (time-invariant) event likelihood Φ (left), and the proportion of time that some agent occupied each subregion during the a simulated mission after significant time has passed (10000 units) (right).

Further, during the simulation, no two agents ever occupied the same space due to the careful parameter manipulations employed by Algorithm 7. Figure 4.6 illustrates the logic governing these manipulations through a simplistic example: During the first update, the blue agent acquires some of the red agent's coverage region. Rather than immediately adding these regions to its active covering, the blue region waits until suf-

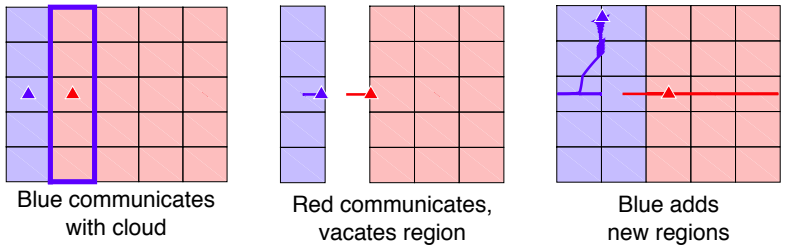

Figure 4.6: Simplified example illustrating how Algorithm 7 manipulates timing parameters to prevent agent collisions: After the blue agent communicates with the cloud, it waits for some amount of time before entering the newly acquired region. During this waiting period, the red agent has time to safely vacate.

ficient time has passed to guarantee that the red agent has updated and moved out of the reassigned regions. Under Algorithm 9, once the red agent communicates with the cloud, it immediately vacates the re-assigned regions, after which the blue agent can add the region to its active covering. This procedure guarantees that no two agents will ever have overlapping active coverings and thus the agents will never collide (Theorem 5). This same logic results in inherent collision prevention over more complex scenarios.

We can also compare the coverage regions produced by Algorithm 7 to those produced by the partitioning algorithm in [1]. The two algorithms were simulated in parallel, performing updates with the same randomly chosen agent-cloud exchange orderings across the two conditions. The left and the right plots in Figure 4.7 show the mean coverage cost over 50 simulation runs, calculated using $\mathcal{H}_{\min}$ (defined in [1], Section II-C) and $\mathcal{H}$ (Section 4.3.2), respectively (portions of the curves extending above the axes indicate an infinite value). The function $\mathcal{H}_{\min}$ is defined nearly identically to $\mathcal{H}$, but uses global graph distances, rather than subgraph distances. It is clear that the evolution produced by the algorithm in [1] converges to a final configuration slightly faster than that produced by Algorithm 7 partitioning whenever costs are quantified using $\mathcal{H}_{\min}$. However, when costs are calculated using $\mathcal{H}$, the algorithms in [1] produced intermediate configurations with infinite cost, indicating disconnected regions, while Algorithm 7 maintained

connectivity. In contrast to [1], our surveillance framework allows for complete coverage without requiring the agents to leave their assigned regions, allowing it to operate more effectively within a decomposition-based, multi-agent surveillance scheme.

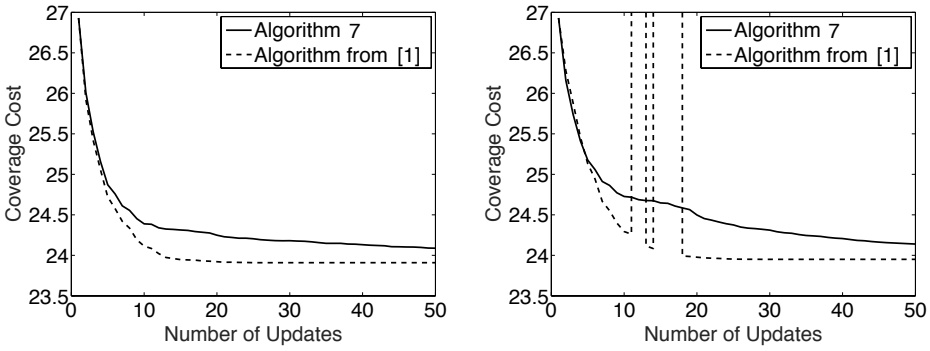

Figure 4.7: Comparison of coverage cost between [1] and Algorithm 7. Coverage costs are calculated with $\mathcal{H}_{min}$ ([1], Section II-C) on the left and with $\mathcal{H}$ (Section 4.3.2) on right in the 4 agent simulated sample mission.

4.5.2 Time-Varying Likelihood

We now illustrate how the proposed coverage framework reacts to changes in the underlying likelihood. Specifically, we study a particular type of time-varying likelihood in which the spatial distribution only changes at discrete time-points, i.e., $\Phi(k, \cdot)$ is piece-wise constant for any $k \in V$. This type of scenario is common in realistic missions, e.g., when the cloud's estimate of the global likelihood is only re-formulated if some agent's sensor data indicates a drastic change in the underlying landscape. For this purpose, we adopt identical parameters as in the first example, with the exception of the likelihood Φ, whose spatial distribution abruptly switches at select time-points. If the switches are sufficiently spaced in comparison to the rate of convergence, then the coverage regions dynamically adjust to an optimal configuration that is reflective of the current state. For example, Figure 4.9 shows the coverage region evolution after the underlying likelihood

undergoes a single switch between the likelihoods in Figure 4.8 at time $t = 2000$.

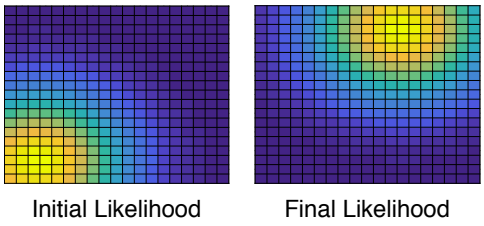

Figure 4.8: The initial and final likelihood $\Phi(\cdot, t)$ for the sample mission with time–varying density (Figure 4.9).

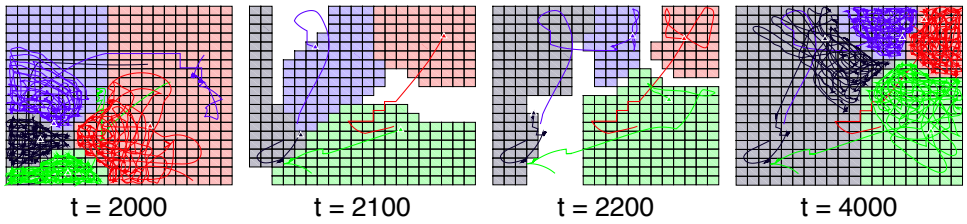

Figure 4.9: Coverage regions after the likelihood switches (see Figure 4.8) during the simulated sample mission.

In contrast, if the likelihood changes faster than the rate of convergence, coverage regions are constantly in a transient state. Despite this, the proposed framework still provides some degree of load-balancing. To illustrate, the left plot in Figure 4.10 shows the value of $\mathcal{H}$ during a simulation in which the underlying likelihood switches at 12 randomly chosen time-points over a 1000 unit horizon. Each switch re-defined the spatial likelihood as a Gaussian distribution centered at a randomly selected location. Notice that the cost is non-increasing between the abrupt spikes caused by changes in the underlying likelihood. A convergent state is never reached; however, coverage regions quickly shift away from high-cost configurations, as seen in the right plot of Figure 4.10, which shows the average percentage drop in the value of the cost $\mathcal{H}$ as a function of the number of non-trivial updates, i.e., updates that did not execute of Algorithm 7, line 3, following an abrupt switch in the likelihood. The percentage drop is calculated with respect to the cost immediately following the most recent switch. During the first nontrivial update,

the cost drops on average 21.8% of the initial post-switch value, indicating a quick shift away from high-cost configurations.

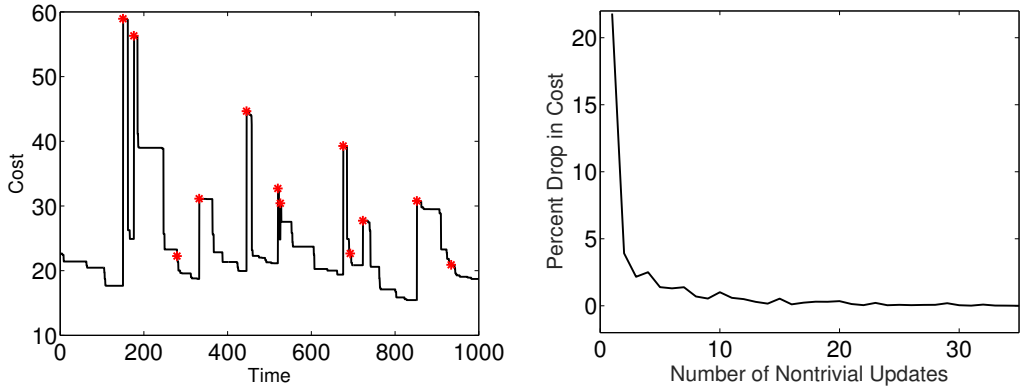

Figure 4.10: Evolution of the cost $\mathcal{H}$ using a piecewise-constant likelihood with 12 random switches (indicated by the stars)(left), and the average percent decrease in $\mathcal{H}$ following each switch (right).

4.6 Chapter Summary

Robotic coverage control problems often play a key role in supervisory control systems. This chapter discusses cloud-supported, decomposition-based, coverage control framework for multi-agent surveillance. In particular, a dynamic partitioning strategy balances the surveillance load across available agents, requiring only sporadic and unplanned agent-cloud exchanges. The partitioning update algorithm also manages high-level logic parameters to guarantee that the resulting coverage assignments have geometric and temporal properties that are amenable for combination with generic single vehicle trajectory planners. In certain cases, the proposed algorithms produce a Pareto optimal configuration, while ensuring collision avoidance throughout.

Future work should further relax communication assumptions to reflect additional limitations, e.g., directional antennae for wireless transmission. Other areas of future

research include the combination of peer-to-peer and cloud-based communication, performance comparisons between specific trajectory planners when used within our framework, e.g., those involving Markov chains, and further theoretical performance characterization.

Chapter 5

UAV Surveillance Under Visibility and Dwell-Time Constraints

Continuing our exploration into intelligent, sensor-focused coordination schemes, this chapter considers a different type of multi-agent mission that involves persistent surveillance of static, discrete points, rather than planar regions (as in the previous chapter). This scenario typically arises in missions where key targets or points of interest have known locations, and thus exploration of large geographic areas is not necessary. For example, if a particular target of interest is known to reside within one of a set of 3 buildings, the mission planner may wish to use the autonomous sensors to provide continuous surveillance of those particular geographic locations. If the number of available agents is smaller than the number of discrete points of interest, this brings about a difficult combinatorial routing problem to determine the manner in which the targets should be visited. To further complicate matters, realistic surveillance missions often also possess additional routing constraints and multiple, conflicting performance objectives.

This chapter considers one such multi-objective, discrete surveillance problem that is motivated by supervisory missions involving fixed-wing UAVs, in which the vehicles

are required to visit a set of targets for the purpose of collecting and transmitting visual imagery to a remotely located human operator for analysis. We assume that particular imaging behaviors are required at each target, e.g., specific camera angles or views, and that, due to the need to transmit real-time visual imagery to the operator, the UAVs are required to dwell at each target for some amount of time. The ideal UAV routes would allow all viewing and dwell-time constraints to be satisfied, while simultaneously minimizing both the total amount of time to visit all of the targets and the time required to reach the first target. Here, the presence of multiple objectives and various imaging constraints makes existing methods for similar combinatorial routing problems ill-suited for use in this domain. In response, this chapter develops novel heuristics for generating high-quality UAV routes within a simple, practical, and modular framework.

We note that this chapter focuses exclusively on coordinating autonomous agents, and thus does not explicitly consider operator behavior. Extensions that directly include the operator within a joint optimization framework are considered in Chapter 6.

5.1 Problem Formulation

We start by rigorously formulating the discrete surveillance problem for a single UAV mission. We then extend to the multi-UAV case in Section 5.5.

5.1.1 UAV Specifications

Consider a single fixed-wing UAV, equipped with a GPS location device and a gimbaled, omnidirectional camera. The camera is steered by a low-level controller, which is independent of the vehicle motion controller. For simplicity, neglect the possibility of camera occlusions that are brought about by vehicle motion (usually not a restrictive assumption, provided the vehicles fly at sufficiently high altitudes and reasonable target

imaging specifications are given). The work herein focuses on high-level UAV trajectory planning, rather than low-level vehicle or camera motion control. We consider planar motion in a global, ground-plane reference frame, assuming the UAV maintains a fixed altitude a and a fixed speed s. We model the UAV as a Dubins vehicle [154] with minimum turning radius r, neglecting dynamic effects caused by wind, e.g. drift or drag. Let $v_0 \in \mathbb{R}^2 \times [0, 2\pi)$ denote the UAV's initial configuration (location, heading).

5.1.2 Target Specifications

Consider $M \in \mathbb{N}$ static targets, each with associated imaging, i.e., visibility and dwell-time, requirements. Each target[1] T_j is associated with the following (fixed) parameters:

1. $t_j \in \mathbb{R}^2$, the location of the target in the ground-plane reference frame,
2. $\texttt{BEH}_j \in \{\texttt{ANY}, \texttt{ANGLE}, \texttt{FULL}\}$, the required viewing behavior, where `ANY` indicates no preference for the azimuth of the collected images, `ANGLE` indicates that the target should be imaged at a specific azimuth, and `FULL` indicates that a 360-degree view of the target should be provided,
3. $\left[\phi_j^{\text{Az}} - \Delta_j^{\text{Az}}, \phi_j^{\text{Az}} + \Delta_j^{\text{Az}}\right] \subset \mathbb{R}$, a range of acceptable azimuths when $\texttt{BEH}_j = \texttt{ANGLE}$, as measured with respect to a reference ray in the ground plane (Figure 5.1, left),
4. $\left[\phi_j^{\text{T}} - \Delta_j^{\text{T}}, \phi_j^{\text{T}} + \Delta_j^{\text{T}}\right] \subset (0, \frac{\pi}{2}]$, acceptable camera tilt angles as measured with respect to a plane parallel to the ground-plane (Figure 5.1, right), and
5. $\tau_j \in \mathbb{Z}_{\geq 0}$, the required number of dwell-time "loops."

[1]Notice that we have used the symbol T_j to denote the j-th target, whereas in Chapter 3 the symbol T_j denoted a task to be completed by the operator. This usage is strategic, since, in the motivating supervisory mission, each target corresponds to an image analysis task to be completed by the operator. This relationship is explored more explicitly in Chapter 6.

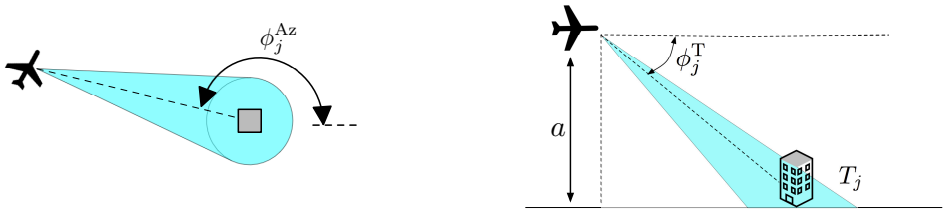

Figure 5.1: An illustration of key imaging parameters associated target T_j. Parameters are measured with respect to a fixed, global reference frame.

Figure 5.2: Example visibility region $\texttt{VIS}_j$ (green shaded area) associated with some target T_j when $\texttt{BEH}_j \neq \texttt{ANGLE}$ (left), and when $\texttt{BEH}_j = \texttt{ANGLE}$ (right). Notice that the visibility region forms either a full annulus or an annular sector in the ground plane.

Define the *visibility region*, $\texttt{VIS}_j \subset \mathbb{R}^2$, for target T_j as the set of locations from which the UAV is able to image the target with an acceptable tilt angle and azimuth (that is, a tilt angle within the interval $\left[\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}}, \phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}}\right]$ and, if $\texttt{BEH}_j = \texttt{ANGLE}$, an azimuth within the interval $\left[\phi_j^{\mathrm{Az}} - \Delta_j^{\mathrm{Az}}, \phi_j^{\mathrm{Az}} + \Delta_j^{\mathrm{Az}}\right]$; if $\texttt{BEH}_j \neq \texttt{ANGLE}$, then any azimuth is acceptable). Each $\texttt{VIS}_j$ is uniquely defined by the UAV altitude a, the location t_j, the behavior $\texttt{BEH}_j$, and the intervals $\left[\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}}, \phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}}\right]$, $\left[\phi_j^{\mathrm{Az}} - \Delta_j^{\mathrm{Az}}, \phi_j^{\mathrm{Az}} + \Delta_j^{\mathrm{Az}}\right]$. Algorithm 10 presents the methodology for constructing visibility regions. Notice that, if $\texttt{BEH}_j \neq \texttt{ANGLE}$, then $\texttt{VIS}_j$ is a full annulus centered at t_j; otherwise, $\texttt{VIS}_j$ is an annular sector (Figure 5.2). As such, fixing target locations, each visibility region is parameterized by two radii (lower and upper limits) together with two angles (lower and upper angular limits).

Algorithm 10: *Visibility Region Construction*

Input : a; $\phi_j^{\mathrm{T}}, \phi_j^{\mathrm{Az}}, \Delta_j^{\mathrm{T}}, \Delta_j^{\mathrm{Az}}$ for each $j \in \{1, \ldots, M\}$
Output : $\{\texttt{VIS}_j\}_{j\in\{1,\ldots,M\}}$

for *Each* T_j **do**
  if $\texttt{BEH}_j \neq \texttt{ANGLE}$ **then**
1 Define $\texttt{VIS}_j$ as the annulus in $\mathbb{R}^2$ centered at t_j with lower, upper radial limits $a/\tan(\phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}})$, $a/\tan(\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}})$, resp.
  else
2 Define $\texttt{VIS}_j$ as the annular sector in $\mathbb{R}^2$ centered at t_j with lower, upper radial limits $a/\tan(\phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}})$, $a/\tan(\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}})$, resp., and lower, upper angular limits equivalent to $\phi_j^{\mathrm{Az}} - \Delta_j^{\mathrm{Az}}$, $\phi_j^{\mathrm{Az}} + \Delta_j^{\mathrm{Az}}$, resp.

3 **return** $\{\texttt{VIS}_j\}_{j\in\{1,\ldots,M\}}$

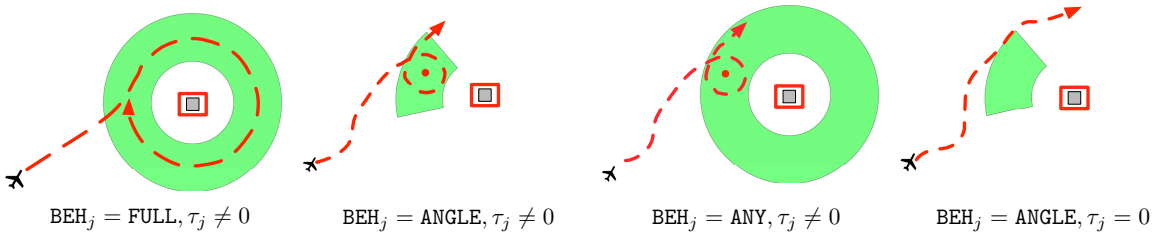

Figure 5.3: Example imaging behaviors at target T_j for various choices of $\texttt{BEH}_j$ and τ_j. The cases where $\tau_j = 0$ and $\texttt{BEH}_j \in \{\texttt{ANY}, \texttt{FULL}\}$ are very similar to the $\tau_j = 0$, $\texttt{BEH}_j = \texttt{ANGLE}$ case, and are thus omitted from the illustration.

The variable[2] $\tau_j \in \mathbb{Z}_{\geq 0}$ indicates the number of dwell-time "loops" that are required at the target T_j. If $\tau_j = 0$, then the UAV is only required to pass over any point within $\texttt{VIS}_j$. If $\tau_j > 0$, i.e., non-trivial dwell time is specified, assume the UAV images T_j as follows: If $\texttt{BEH}_j = \texttt{FULL}$, the UAV makes τ_j full circles around the target location at some constant radius; if $\texttt{BEH}_j \neq \texttt{FULL}$, then the UAV selects a pivot point within $\texttt{VIS}_j$ and makes τ_j circles about the selected point at radius r. Each non-trivial dwell-time maneuver must be performed entirely within the appropriate visibility region. Figure 5.3 shows example imaging behavior for various choices of τ_j and $\texttt{BEH}_j$. For the remaining analysis, assume that imaging parameters are chosen to ensure problem feasibility, i.e.,

[2]Once again, since, in the motivating supervisory application, the dwell-time required at a target corresponds to the time required by the human operator for image analysis, the use of τ_j here to represent the number of dwell-time loops required and the use of τ_j in Chapter 3 to represent a task processing time is strategic. This is explored further in Chapter 6.

there exists at least 1 valid dwell-time maneuver at each target satisfying the required constraints.

Remark 14 (Feasibility) *Feasibility can always be achieved by choosing tolerance parameters sufficiently large.*

5.1.3 Problem Statement

The goal is to construct an optimal UAV trajectory with the following characteristics: The UAV begins its tour by moving from its initial configuration v_0 to a configuration where it can begin imaging a target and, after the initial maneuver, the UAV follows a closed trajectory, along which it images each target to specification. By separating the initial maneuver from the remaining closed route, we ensure that imaging behaviors can be effectively repeated if necessary (since the UAV finishes the closed portion of the tour at the initial imaging location, rather than unnecessarily returning to v_0). Recall the metrics to be minimized: (i) the time required for the UAV to traverse the closed portion of its trajectory (beginning/ending at the first target), and (ii) the time required for the UAV to perform its initial maneuver, i.e., move from v_0 to the starting point of the closed portion. Since the metrics are conflicting in general, a tradeoff must be made in formulating the optimization problem. Consider the following generic formulation.

Problem 1 (Optimal UAV Tour) *Find a UAV tour (consisting of an initial maneuver, and a closed trajectory) that solves the following optimization problem:*

$$\begin{aligned} \textit{Minimize:} \quad & \textit{Closed Trajectory Time} \\ \textit{Subject To:} \quad & \textit{Initial Maneuver Time} \leq \epsilon \\ & \textit{UAV Dynamic Constraints Satisfied (Section 5.1.1)} \\ & \textit{Correct Dwell-Time Maneuvers Performed at Each Target (Section 5.1.2)}, \end{aligned} \tag{5.1}$$

where $\epsilon \geq 0$ is a fixed parameter.

Remark 15 (Scalarization) *The problem* (5.1) *is an ϵ-constraint scalarization of the multi-objective problem (see Section 1.2). A typical alternative scalarization would instead account for the initial maneuver time within a linear objective: α (Initial Maneuver Time)+ β (Closed Trajectory Time) where α, β are constant parameters. For fixed α, β, any optimal solution to an instance of the alternative formulation is also an optimal solution to an instance of* (5.1) *for some choice of ϵ. Since parameter selection and subsequent solution of the linear alternative typically requires construction of a Pareto optimal front by solving instances of* (5.1)*, we restrict our attention to the ϵ constraint scalarization.*

Remark 16 (Relation to [124]) *If (i) the parameter ϵ is sufficiently large (the initial maneuver is inconsequential), and (ii) $\tau_j = 0$ for all j (dwell-times are trivial), then solving Problem 1 is equivalent to solving a* Polygon-Visiting Dubins Traveling Salesperson Problem*, as in [124]. Our methods are loosely based on [124], though we consider a more general multi-objective framework that also incorporates non-trivial dwell-times.*

5.2 Discrete Approximation

Problem 1, which explicitly considers all target imaging constraints, has an infinite number of potential solutions and is difficult to solve. However, by carefully sampling the UAV configuration space, we can pose a finite, discrete alternative whose optimal solutions approximate those of the original problem. The discrete approximation is still a combinatorial search; however, it is closely related to standard path-finding problems, allowing us to leverage existing solvers to produce high-quality sub-optimal solutions. This section develops the discrete approximation of interest.

5.2.1 Configuration Space Sampling

Recalling the Dubins vehicle model, we sample the UAV configuration space to obtain a finite collection of points of the form $v \coloneqq (x, \theta) \in \mathbb{R}^2 \times [0, 2\pi)$. These points serve as the basis for the discrete approximation to Problem 1. Specifically, we sample points that each represent the starting and ending configuration of an appropriate dwell-time "loop" at some target (valid dwell-time maneuvers each start and end at the same configuration). That is, each sampled point $v \coloneqq (x, \theta)$ has heading θ that points in a direction tangent to a valid dwell-time loop (associated with some target $\texttt{TAR}_v$) passing through the location $x \in \texttt{VIS}_j$ (Remark 17). By explicitly pairing each v with its target $\texttt{TAR}_v$, this procedure creates a one-to-one mapping between the generated points and a set of feasible dwell-time maneuvers. As such, subsequent graph formulations can "disregard" dwell-time constraints by using an augmented distance metric. Figure 5.4 shows examples of valid sampled sets associated with some T_j, for various $\texttt{BEH}_j$ and τ_j values.

Algorithm 11 outlines the sampling process. Here, each set $\texttt{DWL}_j$ is defined thusly: If $\tau_j \neq 0$, let $\texttt{DWL}_j$ be the set of points $v \coloneqq (x, \theta) \in \texttt{VIS}_j \times [0, 2\pi)$ having location x that lies on the circular image of an appropriate dwell-time maneuver and heading θ that points in

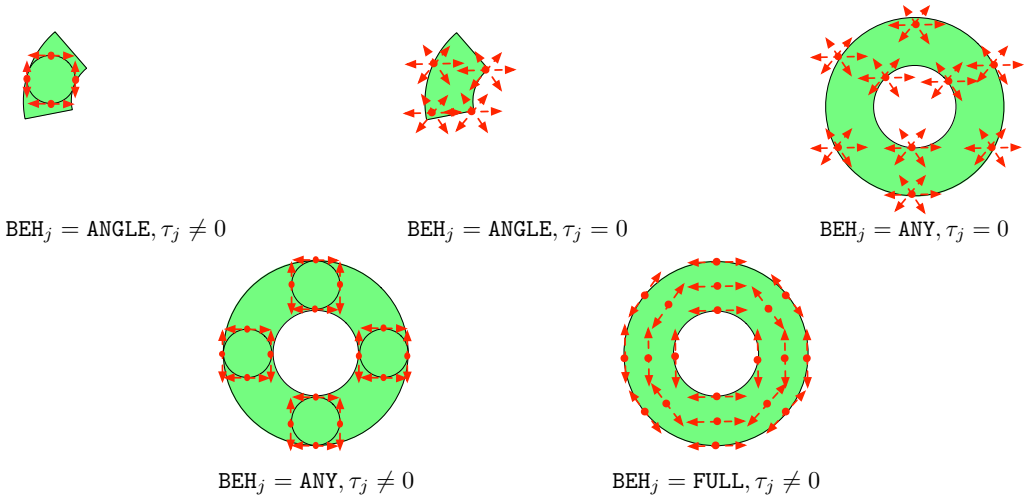

Figure 5.4: Examples of valid configuration samples associated with the target T_j for various choices of $\texttt{BEH}_j$ and τ_j. Here, the red dot is the sampled point's location and the arrow represents its heading (distinct points can have the same planar location). Notice that each discrete point has a location and heading that represents the beginning and ending configuration of a valid dwell-time loop at the target T_j.

a direction tangent to the same circular image at x. If $\tau_j = 0$, let $\texttt{DWL}_j \coloneqq \texttt{VIS}_j \times [0, 2\pi)$. Notice Algorithm 11 allows multiple "copies" of the same configuration be sampled, provided each is associated with a distinct target, i.e., there may exist $v_{k_1}, v_{k_2} \in V$ with identical locations and headings so long as $\texttt{TAR}_{v_{k_1}} \neq \texttt{TAR}_{v_{k_2}}$.

Algorithm 11: *Configuration Space Sampling*

Input : $N_s \in \mathbb{N}$; a; $\texttt{VIS}_j$, τ_j for all $j \in \{1, \ldots, M\}$
Output : V, $\{\texttt{TAR}_v\}_{v \in V}$

Initialize $V = \emptyset$;
for *Each* T_j **do**
  Construct and parameterize $\texttt{DWL}_j$ by considering images of valid dwell-time maneuvers at T_j;
  for $k \in \{1, \ldots, N_s\}$ **do**
    Sample $v_k \in \texttt{DWL}_j$, associate the target T_j to v_k (define $\texttt{TAR}_{v_k} \coloneqq T_j$), and add v_k to V;
return V, $\{\texttt{TAR}_v\}_{v \in V}$

Remark 17 (Dwell-Time Loops) *Under the sampling scheme in Algorithm 11, it is possible that some $v \in V$ is tangent to multiple, distinct dwell-time loops associated with $\texttt{TAR}_v$. Notice that all such loops have identical radii, i.e., the UAV requires the same*

amount of time to traverse each. Thus, we can assume without loss of generality that each v is the starting and ending configuration of a single loop associated with $\texttt{TAR}_v$.

5.2.2 Graph Construction

Algorithm 11 returns a discrete set V, where each individual configuration $v \in V$ represents a point in the UAV configuration space $\mathbb{R}^2 \times [0, 2\pi)$ that is the starting and ending point of a valid dwell-time maneuver at its associated target $\texttt{TAR}_v$. Recalling that the optimal Dubins path between any two configurations $v_{k_1}, v_{k_2} \in \mathbb{R}^2 \times [0, 2\pi)$ is well defined (and easily computed) [154], we can utilize Algorithm 12 to construct a weighted, directed graph $G \coloneqq (V, E, W)$ that effectively discretizes the feasible region of Problem 1. Here, the edge set E contains directed edges connecting each pair of nodes in V that are associated with distinct targets, along with directed edges connecting the initial UAV configuration v_0 with each node in V. Weights are defined via an augmented Dubins distance that includes both the time required to complete the dwell-time maneuver at the source node and the time required to travel between configurations (recall that optimal Dubins paths are asymmetric in general). We are now ready to formally define the discrete approximation to Problem 1 using the graph G.

Problem 2 (Discrete Approximation) *Consider the graph $G \coloneqq (V, E, W)$ resulting from Algorithm 12. Find a sequence $v_1, v_2, \ldots, v_M \in V$ that solves*

$$\begin{aligned} \textit{Minimize: } & W(v_M, v_1) + \sum_{k=1}^{M-1} W\left(v_k, v_{k+1}\right) \\ \textit{Subject To: } & W\left(v_0, v_1\right) \leq \epsilon \\ & \texttt{TAR}_{v_{k_1}} \neq \texttt{TAR}_{v_{k_2}}, \textit{ for any } k_1 \neq k_2 \end{aligned} \tag{5.2}$$

where $v_0 \in V$ corresponds to the initial UAV location and $\epsilon \geq 0$ is a constant parameter.

Algorithm 12: *Graph Construction*

Input : V, $\{\texttt{TAR}_v\}_{v \in V}$; v_0, s, a, r
Output : G

Initialize the edge set $E = \emptyset$;
for *Each pair of distinct points* $v_{k_1}, v_{k_1} \in V$ **do**
    Add the directed edges (v_{k_1}, v_{k_2}) and (v_{k_2}, v_{k_1}) to E;
    Set the weight $W(v_{k_1}, v_{k_2})$ equal to the sum of:
        (i) the time required to perform the dwell-time maneuver associated with v_{k_1}, and
        (ii) the time required to traverse the optimal Dubins path from v_{k_1} to v_{k_2};
    Set the weight $W(v_{k_2}, v_{k_1})$ equal to the sum of:
        (i) the time required to perform the dwell-time maneuver associated with v_{k_2}, and
        (ii) the time required to traverse the optimal Dubins path from v_{k_2} to v_{k_1};
Add the UAV's initial configuration v_0 to V;
for *Each node* $v \in V$ **do**
    Add the directed edges (v_0, v) to E;
    Set $W(v_0, v)$ equal to the time required to traverse the optimal Dubins path from v_0 to v;
return $G = (V, E, W)$

5.3 UAV Tour Construction

Feasible solutions to Problem 1 can be recovered from feasible solutions to Problem 2. Indeed, given a feasible solution $v_1, \ldots, v_M$ to (5.2), we recover a feasible solution to (5.1) by: (i) concatenating the optimal Dubins paths between adjacent nodes in the sequence (appending the path from v_0 to v_1 at the start and the path from v_M with v_1 at the end) and (ii) appending the dwell-time trajectory associated to each node $v_1, \ldots, v_M$. The remainder of our analysis studies the discrete approximation (Problem 2) and its relation to the non-discrete analog (Problem 1).

5.3.1 Solving the Discrete Problem

We leverage solutions of a classic graph path-finding problem to construct solutions to (5.2). In particular, we propose a heuristic framework that relates solutions of (5.2)

to those of an (asymmetric) *Generalized Traveling Salesperson Problem* (GTSP) (see Section 1.2), which is defined for convenience here.

Problem 3 (GTSP) *Given a complete, weighted, directed graph*[3] $G(\mathcal{V}) \coloneqq (\mathcal{V}, \mathcal{E}, \mathcal{W})$, *and a family of finite, non-empty subsets* $\{\mathcal{V}_j \subseteq \mathcal{V}\}_{j\in\{1,\dots,M\}}$, *find a minimum weight, closed path within* $G(\mathcal{V})$ *that visits exactly one node from each subset* $\mathcal{V}_j$.

Remark 18 (GTSP Formulation) *A common alternative GTSP formulation requires the closed path to visit* at least *one node from each* $\mathcal{V}_j$, *rather than* exactly *one node from each* $\mathcal{V}_j$ *as in Problem 3. However, if edge weights satisfy a triangle inequality, then this alternative and Problem 3 are identical. In what follows, we define GTSP instances on an induced subgraph* $G(\mathcal{V}) \subseteq G$, *where* G *is the graph constructed in Algorithm 12. In this case, edge weights in* $G(\mathcal{V})$ *represent an augmented Dubins distance and, since the Dubins distance function satisfies a triangle inequality [125], edge weights in* $G(\mathcal{V})$ *also satisfy a triangle inequality. Thus, we consider Problem 3 without loss of generality.*

Remark 19 (GTSP Solutions) *The standard GTSP is NP-hard. However, practical strategies exist for quickly constructing high-quality solutions, e.g., transformation of the problem into a standard ATSP and application of a heuristic solver (see Section 1.2).*

Note that, in general, Problem 2 is *not* equivalent to a GTSP, due to the constraint on the initial maneuver. Despite this fact, we can still leverage GTSP solution procedures in constructing solutions to the constrained problem. Indeed, a heuristic procedure for constructing solutions to Problem 2 using the solutions to related GTSP instances is outlined in Algorithm 13. Here, $\texttt{INL}_\epsilon$ denotes the set of all nodes in $V\backslash\{v_0\}$ that can be reached from v_0 in time less than ϵ.

[3]We use the symbol $G(\mathcal{V})$ here strategically, since all GTSP instances in subsequent algorithms are defined over an induced subgraph of the larger graph G.

Algorithm 13: *Heuristic Solution to Problem 2*

Input : $G = (V, E, W)$, $\{\texttt{TAR}_v\}_{v \in V \setminus \{v_0\}}$
Output : $v_1, \ldots, v_M$

1 Construct the set $\texttt{INL}_\epsilon \coloneqq \{v \in V \setminus \{v_0\} \mid W(v_0, v) \leq \epsilon\}$;
if *$\texttt{INL}_\epsilon$ is empty* **then**
2 **return** "Problem 2 Infeasible"
else
3 Select a subset $\texttt{INL}_\epsilon^* \subseteq \texttt{INL}_\epsilon$, whose elements are all associated with a single target $T_{\hat{\jmath}}$;
4 Construct the subgraph $G(\mathcal{V}) \coloneqq (\mathcal{V}, \mathcal{E}, \mathcal{W})$ of the graph G that is induced by the node set $\mathcal{V}$, where $\mathcal{V} \coloneqq V \setminus (\{v \mid \texttt{TAR}_v = T_{\hat{\jmath}}, v \notin \texttt{INL}_\epsilon^*\} \cup \{v_0\})$;
5 Formulate and construct a solution to the GTSP (Problem 3) using the graph $G(\mathcal{V})$ and subsets $\mathcal{V}_j \coloneqq \{v \in \mathcal{V} \mid \texttt{TAR}_v = T_j\}$;
6 Cyclically permute the GTSP solution to obtain a sequence $v_1, v_2, \ldots, v_M$ with $\texttt{TAR}_{v_1} = T_{\hat{\jmath}}$
7 **return** $v_1, \ldots, v_M$

In general, the sequences produced by Algorithm 13 will not be optimal with respect to Problem 2. They will, however, be feasible. Further, if $\texttt{INL}_\epsilon$ has a particular structure, then the subset $\texttt{INL}_\epsilon^*$ (see Algorithm 13) can be chosen to ensure that the GTSP instance (line 5) is equivalent to Problem 2. The following results make this discussion precise.

Theorem 6 (Feasibility) *Algorithm 13 produces a feasible solution to Problem 2.*

Proof: The GTSP solution (line 5) will contain some $v \in \texttt{INL}_\epsilon^* \subseteq \texttt{INL}_\epsilon$. Thus, the permutation operation in line 6 will produce $v_1, v_2, \ldots, v_M$ with $v_1 \in \texttt{INL}_\epsilon$. It follows readily that the algorithmic output is feasible with respect to Problem 2 ■

Remark 20 (Feasibility) *The feasibility result of Theorem 6 does not require the construction of an optimal GTSP solution; that is, the theorem result holds provided that* any *feasible GTSP solution is produced in line* 5*.*

Theorem 7 (Equivalence) *Consider Algorithm 13. Suppose $\texttt{INL}_\epsilon$ is nonempty and that there exists an index $\hat{\jmath} \in \{1, \ldots, M\}$ satisfying either:*

1. *all nodes in $\texttt{INL}_\epsilon$ are associated with target $T_{\hat{\jmath}}$, or*

2. *all nodes in V that are associated with $T_{\hat{\jmath}}$ belong to $\texttt{INL}_\epsilon$.*

Then, if $\texttt{INL}_\epsilon^*$ *(line 3) is chosen as the set of all nodes in* $\texttt{INL}_\epsilon$ *that are associated with some such* $T_{\hat{\jmath}}$*, then optimal solutions of the GTSP in line 5 map to those of Problem 2 via the operation in line* 6*. That is, if a globally optimal solution to the GTSP is produced, then the output* $v_1, \ldots, v_M$ *of Algorithm 13 is a globally optimal solution to Problem 2.*

Proof: Any feasible solution $v_1, v_2, \ldots, v_M$ to Problem 2 must contain exactly one node associated to each target, where $v_1 \in \texttt{INL}_\epsilon$. If $\texttt{INL}_\epsilon$ contains only nodes associated to $T_{\hat{\jmath}}$ (condition (i)), then no feasible solution to Problem 2 contains any $v \notin \texttt{INL}_\epsilon$ with $\texttt{TAR}_v = T_{\hat{\jmath}}$. Thus, there is no loss of generality in considering the modified graph $G(\mathcal{V})$ when $\texttt{INL}_\epsilon^* = \texttt{INL}_\epsilon$. The same applies when $\texttt{INL}_\epsilon$ satisfies condition (ii) and $\texttt{INL}_\epsilon^*$ equals the set of all nodes associated with $T_{\hat{\jmath}}$, as this implies $G(\mathcal{V}) = G(V \backslash \{v_0\})$. Since cyclic permutation of the node sequence does not affect the cost, any optimal solution to the GTSP in line 5 maps to an optimal solution of Problem 2 via the operation in line 6. ■

Figure 5.5 shows a graphical illustration of the theorem conditions 1 and 2. We note that the conditions required by Theorem 7 are frequently met when target spacing is large, a condition that is common in realistic supervisory surveillance missions.

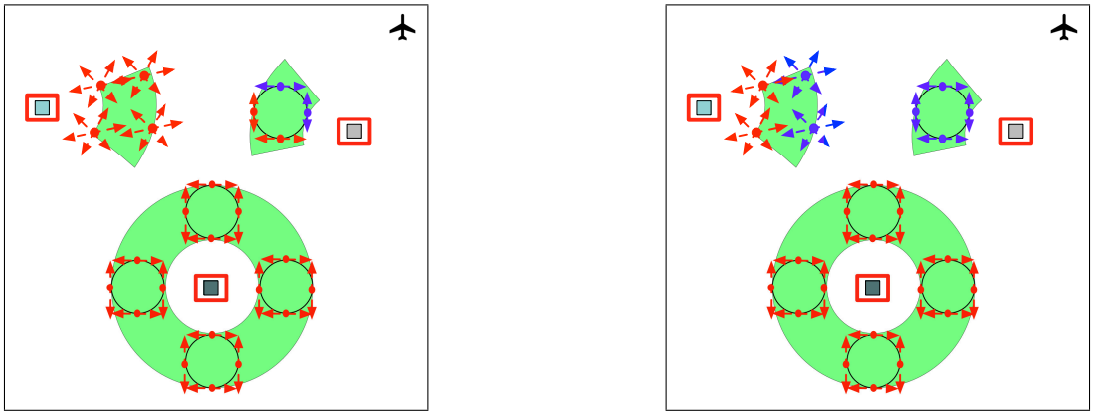

Figure 5.5: Diagrams illustrating two cases when $\texttt{INL}_\epsilon$ (blue nodes), i.e., the set of nodes that the UAV can reach from its initial configuration, satisfies condition 1 (left) and 2 (right) of Theorem 7.

5.3.2 Complete Tour Construction

Algorithm 14 is a heuristic procedure that leverages solutions to the discrete approximation (Problem 2) to construct solutions to the full routing problem (Problem 1). Solutions produced by Algorithm 14 are not optimal in general, though they will exhibit structural characteristics that will generally improve in quality (with respect to Problem 1) as the sampling granularity is made increasingly fine. That is, Algorithm 14 is *resolution complete* in some sense, providing justification for the sampling-based approximation approach. A precise characterization of the resolution completeness properties is somewhat tedious, and thus is postponed until Appendix B.

Algorithm 14: *Heuristic Tour Construction Using GTSPs*

Input : $v_0, s, a, r; N_s; \{T_j\}_{j\in\{1,\dots,M\}}$
Output : Complete UAV Route

% Create visibility regions;
Create target visibility regions via Algorithm 10;
% Create the discrete approximation;
Sample the configuration space and create the graph G via Algorithms 11 and 12;
Formulate Problem 2;
% Solve the discrete approximation;
Construct a solution $v_1, \dots, v_M$ to Problem 2 via Algorithm 13;
if *Algorithm 13 returns an error (Problem 2 is infeasible)* **then**
    return "Error: Discrete Approximation Infeasible"
% Convert the solution of Problem 2 into a solution to Problem 1;
Construct the optimal Dubins path that visits the nodes in the following order: $v_0, v_1, \dots, v_M, v_1$;
Recover a feasible solution to Problem 1 by appending dwell-time maneuvers.;
return Complete UAV Tour: Initial Maneuver + Closed Trajectory

5.4 Numerical Examples

We illustrate our algorithms through numerical examples. For each simulated mission, solutions to Problem 1 are constructed via Algorithm 14, where GTSPs are solved through transformation into an equivalent ATSP (see [128]) that is subsequently solved using

Table 5.1: Target Input Data (Pareto-Optimality Study)

T_j	t_j	$\texttt{Beh}_j$	τ_j	$\left[\phi_j^{\mathrm{Az}} - \Delta_j^{\mathrm{Az}}, \phi_j^{\mathrm{Az}} + \Delta_j^{\mathrm{Az}}\right]$	$\left[\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}}, \phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}}\right]$
T_1	$(5000, -5000)$ m	FULL	2	–	$[\frac{\pi}{8}, \frac{3\pi}{8}]$
T_2	$(4300, -1750)$ m	ANGLE	1	$[\frac{\pi}{4}, \frac{3\pi}{4}]$	$[\frac{\pi}{8}, \frac{3\pi}{8}]$
T_3	$(0, 4000)$ m	FULL	3	-	$[\frac{\pi}{8}, \frac{3\pi}{8}]$
T_4	$(-8000, -2000)$ m	ANY	1	-	$[\frac{\pi}{8}, \frac{3\pi}{8}]$
T_5	$(-2000, 8000)$ m	ANGLE	0	$[\frac{3\pi}{2}, 2\pi]$	$[\frac{\pi}{8}, \frac{3\pi}{8}]$

the Lin-Kernighan heuristic (as implemented by LKH [87]). In all cases, the set $\texttt{INL}_\epsilon^*$ (Algorithm 13) is chosen as the set of all points in $\texttt{INL}_\epsilon$ associated with some single target (satisfying Theorem 7 conditions whenever possible). A slightly modified version of Algorithm 11 is used for sampling in which the number of samples, N_s, associated with each target is not fixed *a priori*, but instead is determined by creating a grid of samples within the appropriate sampling subsets. The grid spacing is determined by 3 parameters δr, $\delta\theta$, and $\delta\alpha$, which represent, loosely, the radial location spacing, the angular location spacing, and the angular heading spacing, resp. The parameters δr, $\delta\theta$, and $\delta\alpha$ are inversely proportional to the number of samples at each target, and the sampled set is dense in the limit as spacing parameters jointly tend to 0.

5.4.1 Pareto-Optimal Front

The first example is a 5 target mission with the following UAV parameters: $r = 750$ m, $a = 1000$ m, $s = 39$ m/s, and $v_0 = ((-2500, 500)\text{ m}, 0) \in \mathbb{R}^2 \times [0, 2\pi)$. Target parameters are shown in Table 5.1. The approximate Pareto-optimal front (with respect to ϵ) for Problem 1 as a function of the sampling granularity is shown in Figure 5.6. The figure also shows illustrations of solutions produced at spacing condition 5 when $\epsilon = 65$ s (left) and $\epsilon = 205$ s (right). Recalling that (i) Theorem 7 does not hold for all ϵ, and (ii) heuristic solvers for GTSPs do not guarantee the production of global optima,

Spacing Condition	δr	$\delta\theta$	$\delta\alpha$
1	1000 m	π	π
2	500 m	π	π
3	500 m	$\pi/2$	$\pi/2$
4	250 m	$\pi/2$	$\pi/2$
5	250 m	$\pi/4$	$\pi/4$
6	125 m	$\pi/4$	$\pi/4$
7	125 m	$\pi/8$	$\pi/8$

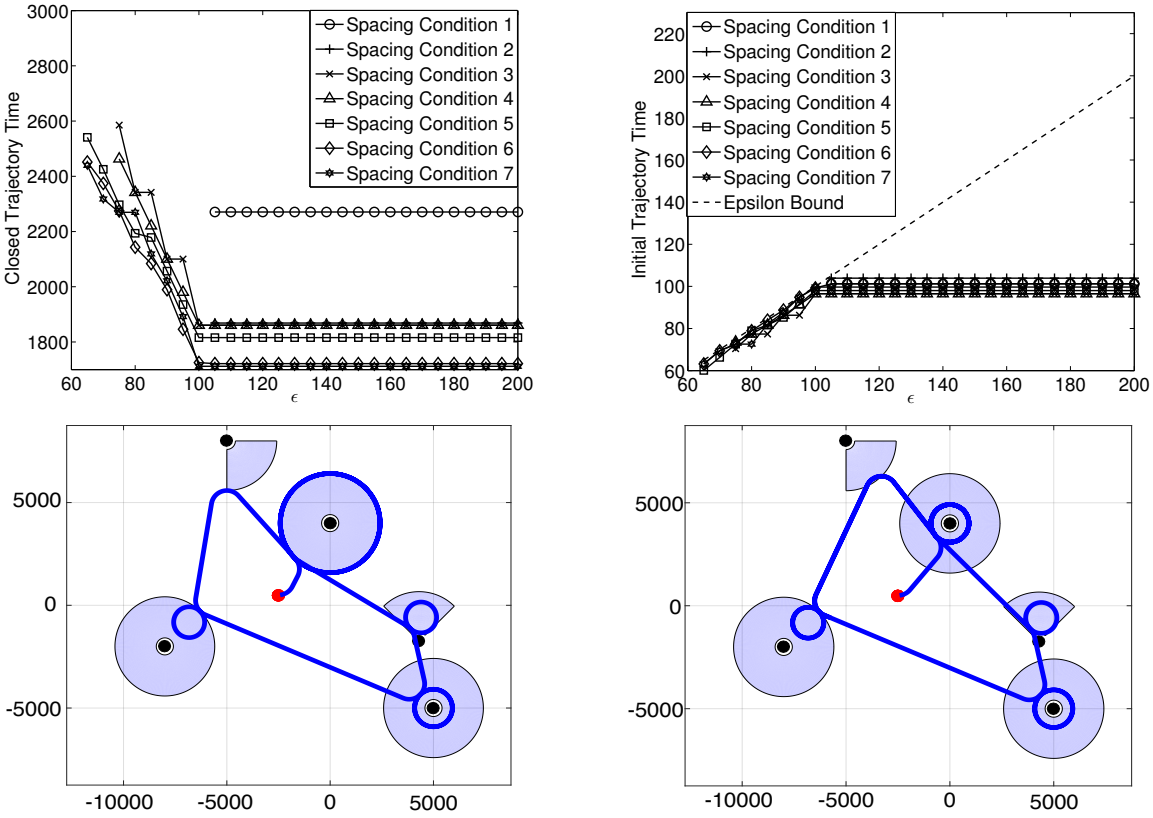

Figure 5.6: Approximate Pareto-optimal front and example routes for a 5-target example mission. The table contains the spacing parameters considered, the middle plots show the closed trajectory times produced (left) and the corresponding initial maneuver times produced (right), and the bottom diagrams shows the optimal routes produced for spacing condition 5 when $\epsilon = 65$ s (left) and $\epsilon = 205$ s (right).

to obtain the best approximation of the Pareto-optimal front, the following steps were taken to generate each curve: First, Algorithm 14 was called for a series of ϵ values, and the resulting initial maneuver/closed trajectory times were recorded. Then, in post-processing, the approximate Pareto-optimal curve was generated by selecting, for each ϵ, the lowest cost route satisfying the initial maneuver constraint out of the solutions produced in the computation stage. Note that increasing ϵ corresponds to relaxing the initial maneuver constraint, and thus the cost is non-increasing in ϵ. Notice also the Pareto-optimal fronts shift toward zero as the sampling spacing is decreased.

Remark 21 (Pareto-optimal Fronts) *The optimal cost of Problem 1 is only sensitive to changes in ϵ over some finite set $\bigcup_{j=1}^{M}[\underline{\epsilon}_j, \overline{\epsilon}_j]$ that depends primarily on target and UAV locations. This structure can be exploited to efficiently compute Pareto-optimal fronts.*

5.4.2 Performance

The next example illustrates the performance of Algorithm 14 in comparison to an incremental, "greedy" alternative that operates as follows: Visibility region creation and configuration space sampling are done using Algorithms 10 and 11. Starting with the initial UAV configuration, each successive UAV destination is chosen by selecting the closest node (Dubins distance, neglecting dwell-times) associated with a target that has not yet been imaged. A valid route is constructed by appending dwell-time maneuvers and connecting the last selected configuration (v_M) with the first (v_1). We consider a 5 target mission with the same UAV parameters and the same target locations, imaging behaviors, and tolerances as in the previous example. However, we vary the number of dwell-time loops associated with each target (though we assume each target requires same number of loops).

The difference between the closed trajectory times produced by the greedy method

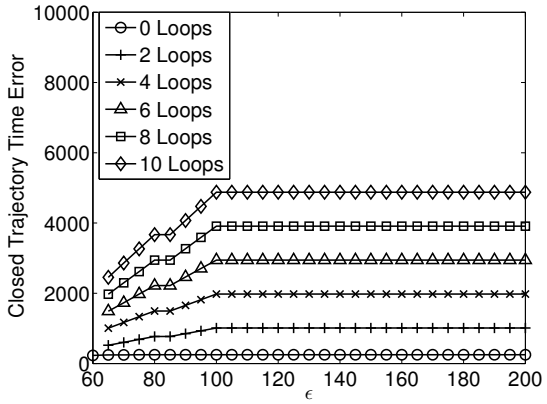

Figure 5.7: Performance of the greedy algorithm with respect to optimal routes. Notice that, when measured as the difference between the total tour lengths, the relative performance of the greedy algorithm in this example can be made arbitrarily poor by increasing the number of dwell-time loops to be performed at each target.

and those produced by Algorithm 14 as a function of ϵ under the spacing condition 5 (Figure 5.6) is shown in Figure 5.7. Notice that the relative performance of Algorithm 14 improves as both ϵ and the number of dwell-time loops at each target are increased. In this example, the performance of the greedy search method can be made arbitrarily poor by increasing the number of dwell-time loops. This result is primarily due to targets that require a 360-degree view, since the greedy heuristic generally chooses those points lying on the perimeter of the visibility region, which are very far from the target location. Thus, increasing the number of dwell-time loops (performed at constant radius) can dramatically increase tour times. As such, Algorithm 14 provides a significant advantage over similar incremental planning strategies when non-trivial dwell-times are required.

5.4.3 Resolution Completeness

To illustrate resolution completeness (see Appendix B), consider a mission with 2 targets whose relevant data is in Table 5.2. The UAV has the same altitude, velocity, and minimum turning radius as the UAVs in the previous examples, but has initial configu-

Table 5.2: Target Input Data (Resolution Completeness Study)

T_j	t_j	$\texttt{Beh}_j$	τ_j	$\left[\phi_j^{\mathrm{Az}} - \Delta_j^{\mathrm{Az}}, \phi_j^{\mathrm{Az}} + \Delta_j^{\mathrm{Az}}\right]$	$\left[\phi_j^{\mathrm{T}} - \Delta_j^{\mathrm{T}}, \phi_j^{\mathrm{T}} + \Delta_j^{\mathrm{T}}\right]$
T_1	$(2131.8, 1026.7)$ m	ANY	0	–	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_2	$(-13840, -5833)$ m	ANY	1	–	$[\frac{\pi}{8}, \frac{3\pi}{8}]$

ration $v_0 = ((0,0)\text{ m}, \pi/7) \in \mathbb{R}^2 \times [0, 2\pi)$. This example has been carefully constructed so that the optimal solution is easily deduced for select ϵ conditions. In particular, when $\epsilon \geq 25$ s, the optimal tour involves the UAV making an immediate left turn at minimum radius, and then visiting the remaining target for a total closed tour length of 848.62 s. When $\epsilon = 16.26$ s, the UAV is constrained to make a shorter initial maneuver, so the UAV travels straight until it hits the visibility region, and then proceeds with the remainder of the tour for an optimal closed tour length of 881.14 s. A schematic showing optimal routes for these two ϵ conditions is shown in the top left portion of Figure 5.8. (the red curve corresponds to the initial turn for the $\epsilon = 16.26$ s case; the remainder of the route is identical). When $\epsilon \geq 25$ s, the problem instance is non-degenerate (Definition 8) and satisfies the conditions required for resolution completeness (Theorem 9). Therefore, if a reasonable sampling method and GTSP solver are used, we expect the solutions produced by Algorithm 14 will tend toward the global optimum with finer sampling. This behavior is illustrated by the top right plot in Figure 5.8, which shows the relative error (difference divided by the optimum) between the cost produced via Algorithm 14 and the globally optimal cost for the sampling conditions listed in the table when $\epsilon = 130$ s. Notice that the relative error tends to zero with finer discretization. Also note that, for this example, the optimal solution to the discrete approximation involved vertices located on visibility region boundaries, resulting in an insensitivity to radial grid spacing. In contrast, when $\epsilon = 16.26$ s, the problem becomes degenerate, since there is a single configuration to which the UAV can travel in order to satisfy the ϵ bound. In this case,

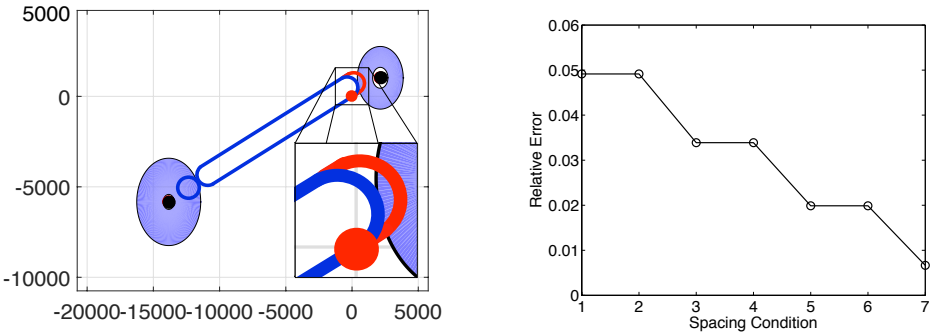

Figure 5.8: Optimal routes when $\epsilon = 16.26$, 25 s (left) and relative cost error when $\epsilon = 130$ s (right) for the example mission described in Section 5.4.3.

Problem 2 is infeasible for any sampling scheme that does not choose configuration in question. Therefore, Algorithm 14 is not resolution complete in the sense of Theorem 9 when $\epsilon = 16.26$. Note, however, that resolution completeness holds if ϵ is increased by any arbitrarily small positive amount.

5.5 Extensions for Multiple Vehicle Missions

Even though the strategies discussed in this chapter thus far are, strictly speaking, only applicable to single vehicle missions, they can easily be paired with a target assignment heuristic in order to address similarly structured multiple-vehicle problems. Indeed, using a *decomposition-based* solution strategy (similar to that discussed in Chapter 4), multiple-vehicle problems can be addressed by first assigning a subset of the targets to each vehicle and subsequently solving each of the resultant single vehicle problems.

We briefly discuss this type of decomposition-based extension here. In addition to their expository value, the following sections also provide the strategies that will be used as a basis of comparison for the joint optimization methods introduced in Chapter 6.

5.5.1 Multi-Vehicle Problem Description

We consider a multiple-vehicle surveillance mission that is analogous to that of Section 5.1. That is, consider a team of $N \in \mathbb{N}$ UAVs, each modeled as in Section 5.1.1, that is responsible for providing surveillance of a set of $M \in \mathbb{N}$ targets, each of which requiring a specific imaging behavior (as in Section 5.1.2). For simplicity, we make the following assumptions: First, suppose that any target is allowed to be imaged by any of the available UAVs, i.e., there is no preference for which particular UAV images a given target. Second, assume that all UAVs are homogenous[4], i.e., all UAVs identically fly at an altitude a with forward airspeed s and have minimum turning radius r. Lastly, we neglect the possibility of UAV collisions in the optimization, i.e., there is no penalty assessed for two UAVs being colocated at some point in time.

Similar to the single-vehicle case, we seek a tour for each UAV (initial maneuver and closed trajectory) that together accomplish the desired imaging behavior at each target. While there are multiple possible formulations of the multi-vehicle problem, for illustrative purposes, we consider minimizing two metrics that are analogous in some sense to those considered in the single vehicle problem: (i) the delay between the mission onset and the first time that one of the UAVs reaches a target, and (ii) total time that it takes to execute 1 iteration of the tour, i.e., the maximum time that it takes any single UAV to complete both its initial maneuver and one iteration of its assigned closed trajectory. This formulation of the multi-vehicle problem is summarized in Problem 4.

Problem 4 (Multi-UAV Tour) *Find a tour for each UAV (consisting of an initial maneuver and a closed trajectory), so that the set of UAV tours collectively solves the*

[4]The methods here readily extend to the heterogenous case with intuitive modifications.

following optimization problem:

$$\begin{aligned} \textit{Minimize:} \quad & \textit{Max Time Required for Any Single UAV to Complete Tour} \\ \textit{Subject To:} \quad & \textit{At least One UAV's Initial Maneuver Time} \leq \epsilon \\ & \textit{UAV Dynamic Constraints Satisfied (Section 5.1.1)} \\ & \textit{Correct Dwell-Time Maneuvers Performed at Each Target (Section 5.1.2)}, \end{aligned} \tag{5.3}$$

where $\epsilon \geq 0$ is a fixed parameter.

As in the single vehicle case, we assume that target parameters are chosen sufficiently large to ensure problem feasibility. Notice that the objective function in (5.3) is subtly different than that of (5.1), since it considers the time to complete the *total* tour, i.e., the sum of the initial maneuver time and the closed trajectory time, rather than just the time required for the closed portion. However, for single-vehicle problems, the two formulations (Problem 1 and Problem 4) are equivalent in some sense when addressing the multi-objective problem: any solution to an instance of Problem 4 is also a solution to an instance of Problem 1 for some (possibly different) choice[5] of ϵ. We choose the formulation of Problem 4 since it directly reflects the goals of many typical multi-UAV surveillance missions. Indeed, the primary performance metric in the multi-vehicle case is usually the time to complete the *entire* mission, rather than simply the closed portion.

5.5.2 Decomposition-Based Solution

A typical decomposition-based solution for Problem 4 is outlined in Algorithm 15. Here, v_0^n represents the initial configuration of the n-th UAV.

[5]This equivalence is similar to the relation between scalarization methods of the muli-objective problem (see Remark 15).

Algorithm 15: *Decomposition-Based Tour Construction Using GTSPs*

Input : $\{v_0^n\}_{n\in\{1,\dots,N\}}, s, a, r; N_s; \{T_j\}_{j\in\{1,\dots,M\}}$
Output : Complete Multi-UAV Route

% Create visibility regions;
1 Create target visibility regions via Algorithm 10;
% Sample configuration space;
2 Sample the configuration space via Algorithm 11;
%Assign Targets to UAVs;
3 Find Target/UAV Pairings;
%Solve individual UAV routing problems;
for *Each UAV n* **do**
4 Create the graph G via Algorithm 12;
5 Choose an appropriate ϵ value;
6 Formulate single-vehicle problem (Problem 1), only considering targets paired to UAV n;
7 Formulate Problem 2 using previously constructed sampling sets;
8 Construct a solution to Problem 2 via Algorithm 13;
if *Algorithm 13 returns an error (Problem 2 is infeasible)* **then**
9 **return** "Error: Discrete Approximation Infeasible";
10 Construct the UAV tour (initial maneuver + closed trajectory) associated with the solution constructed in step 8;
11 **return** UAV Tours

5.5.3 Target Assignment

The only significant algorithmic difference between the decomposition-based, multi-vehicle solution framework of Algorithm 15 and the single vehicle strategy presented in Algorithm 14 is the need to assign targets to the UAVs (line 3). Indeed, after assignment, each of the resultant single vehicle problems are solved as in the previous sections.

This target assignment can be done in a number of ways, and there are efficient, intuitive heuristics that can be used to obtain reasonable performance. However, care must be taken in constructing these heuristics to ensure that Algorithm 15 always produces a feasible solution to Problem 4 whenever such a solution exists. As an example, consider the *greedy* assignment procedure outlined in Algorithm 16. Here, we assume that (i) the variable $v_0^n \in \mathbb{R}^2 \times [0, 2\pi)$ is the initial configuration of the n-th UAV, (ii) visibility regions have been constructed using Algorithm 11 and (iii) the configuration space has

been sampled using Algorithm 11. This greedy strategy operates by sequentially assigning targets to UAVs in a way that locally maximizes the rate of information provided to the user. That is, the algorithm recursively chooses the next UAV-target pairing as the assignment that minimizes the time at which an unserviced target can be reached, given the (theoretical) current system state. Note that this process is purely for assignment, as the single-vehicle protocol of Section 5.3.2 is used to construct final vehicle routes. This construction guarantees that the desired feasibility property of the resultant solution is satisfied whenever ϵ is chosen carefully, as shown by the following Lemma.

Algorithm 16: Target Assignment

Input : a, s, r, $\{v_0^n\}_{n\in\{1,\dots,N\}}$;$\{T_j\}_{j\in\{1,\dots,m\}}$, V, $\{\texttt{TAR}_v\}_{v\in V}$
Output : Target/UAV Pairings

1 Set theoretical UAV positions $\hat{v}^n = v_0^n$ and the timing statistic $\texttt{time}^n = 0$ for all $n \in \{1,\dots,N\}$
2 **while** *There are still unassigned targets* **do**
3 Reset the statistic $t^0 = \infty$.
4 **for** *Each UAV* n **do**
5 **for** *Each vertex* $v \in V$ *associated with an unassigned target* **do**
6 Set `opt` equal to the time required UAV n to traverse the optimal Dubins path between $\hat{v}^n$ and v;
7 **if** $\hat{v}^n \neq v_0^n$ **then**
8 Set `dwell` equal to time required for dwell-time maneuver associated with $\hat{v}^n$.
  else
9 Set $\texttt{dwell} = 0$.
10 **if** $\texttt{time}^n + \texttt{opt} + \texttt{dwell} < t^0$ **then**
11 Set $\hat{\jmath} = \texttt{TAR}_v$, $\hat{n} = n$, $\hat{v} = v$, and $t^0 = \texttt{time}^n + \texttt{opt} + \texttt{dwell}$
12 Assign target $T_{\hat{\jmath}}$ to UAV $\hat{n}$, and set $t^{\hat{n}} = t^0$, $\hat{v}^{\hat{n}} = \hat{v}$;
13 **return** Target/UAV Pairings

Lemma 3 (Multi-Vehicle Feasibility) *Suppose Problem 4 is feasible. If target assignment is performed using Algorithm 16 and, for each UAV, the value of ϵ (line 5) is taken equal to the minimum time required for the UAV to reach any vertex in V associated with one of its assigned targets, then the collection of routes produced by the decomposition-based strategy of Algorithm 15 is a feasible solution to Problem 4.*

Table 5.3: Target Input Data (Multi-Vehicle Study)

T_j	t_j	$\texttt{Beh}_j$	τ_j	$\left[\phi_j^{\text{Az}} - \Delta_j^{\text{Az}}, \phi_j^{\text{Az}} + \Delta_j^{\text{Az}}\right]$	$\left[\phi_j^{\text{T}} - \Delta_j^{\text{T}}, \phi_j^{\text{T}} + \Delta_j^{\text{T}}\right]$
T_1	$(10000, 0)$	FULL	5	–	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_2	$(0, 500)$	ANGLE	1	$[\frac{\pi}{4}, \frac{3\pi}{4}]$	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_3	$(1200, 5000)$	FULL	2	-	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_4	$(-4000, -1500)$	ANY	0	-	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_5	$(-2000, 8000)$	ANGLE	0	$[\frac{3\pi}{2}, 2\pi]$	$[\frac{\pi}{6}, \frac{\pi}{3}]$
T_6	$(3500, -4000)$	ANGLE	5	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[\frac{\pi}{6}, \frac{\pi}{3}]$

Proof: By the heuristic assignment method of Algorithm 16, the first target-UAV pairing will be chosen by minimizing the time for *any* UAV to reach *any* target. Suppose the UAV involved in this pairing is UAV n and the target involved in the pairing is target T_j. If Problem 4 is feasible, then the time that it takes UAV n to reach target T_j is less than the specified constraint in Problem 4. It follows that readily that, if initial maneuvers are constrained to be minimum by appropriate choice of ϵ, then the result of Algorithm 15 is a feasible solution of Problem 4. ■

In addition to ensuring feasibility, the greedy heuristic of Algorithm 16 also tends to provide reasonable performance with respect to the maximum individual UAV tour length when compared to other assignment strategies. To illustrate, consider the following example problem containing 6 targets and 3 homogenous UAVs. In this example, each UAV has a minimum turning radius of 750 m, an altitude of 1000 m, and a constant speed of 39 m/s. The UAV initial configurations are $v_0^1 = ((0, 0)\text{ m}, 0)$, $v_0^2 = ((1000, 0)\text{ m}, 0)$, and $v_0^3 = ((-1000, 0)\text{ m}, 0) \in \mathbb{R}^2 \times [0, 2\pi)$. Relevant data concerning the targets is in Table 5.3. We consider 3 different implementations of the decomposition-based strategy in Algorithm 15, each employing a different target assignment strategy. In the first implementation, the greedy heuristic (Algorithm 16) is used ("Greedy" assignment). In the second implementation, an alternative strategy is used in which each target is paired with the UAV that is closest to the target's physical location according to the Euclidean

distance at the onset of planning ("Closest" assignment). In the third implementation, targets are paired to UAVs randomly via pseudorandom groupings of the set $\{1, \ldots, M\}$ ("Random" assignment). In all cases, the resultant single vehicle problems are solved by choosing ϵ equal to the minimum time required for the UAV in question to reach any node associated with one of its assigned targets, i.e., the initial maneuver is constrained to be as short as possible, given the target-UAV pairings. As in Section 5.4, discretization of visibility regions is performed using a slightly modified version of Algorithm 11 in which the number of discrete samples, N_s, associated with each target is not fixed *a priori*, but instead is determined by creating a grid of samples within the appropriate sampling subsets with spacing that is determined by the 3 fixed parameters δr, $\delta\theta$, and $\delta\alpha$. We let $\delta r = 250$ m, $\delta\theta = \pi/4$, and $\delta\alpha = \pi/4$ in all trials.

Illustrations of the solution produced using a single instance of each of the aforementioned algorithmic implementations is shown in Figure 5.9. In the figure, the top left, top right, and bottom schematics depict the routes constructed using Algorithm 15 in conjunction with the "Greedy," "Closest," and "Random" assignment strategies, resp. Black dots represent target locations, while the shaded annular areas represent visibility regions. Solid lines each represent one UAV tour, with the colored squares marking the initial UAV locations. Figure 5.10 shows a comparison of the maximum individual UAV tour lengths contained in the solutions to each of 2000 simulation runs computed using the 3 implementations of Algorithm 15 ("Greedy," "Closest," and "Random"). The histogram shows the result of each simulation run using the "Random" target assignment strategy, while the dashed lines illustrate the mean maximum individual tour time contained in the solutions produced using the "Greedy" and "Closest" strategies. Note that, in Figure 5.10, we have omitted a histogram of the costs produced using the "Greedy" and "Closest" algorithms for clarity, as there was very little variance in the costs produced for these two alternatives ("Greedy": 95% of trials had a cost of 1694 s, with a

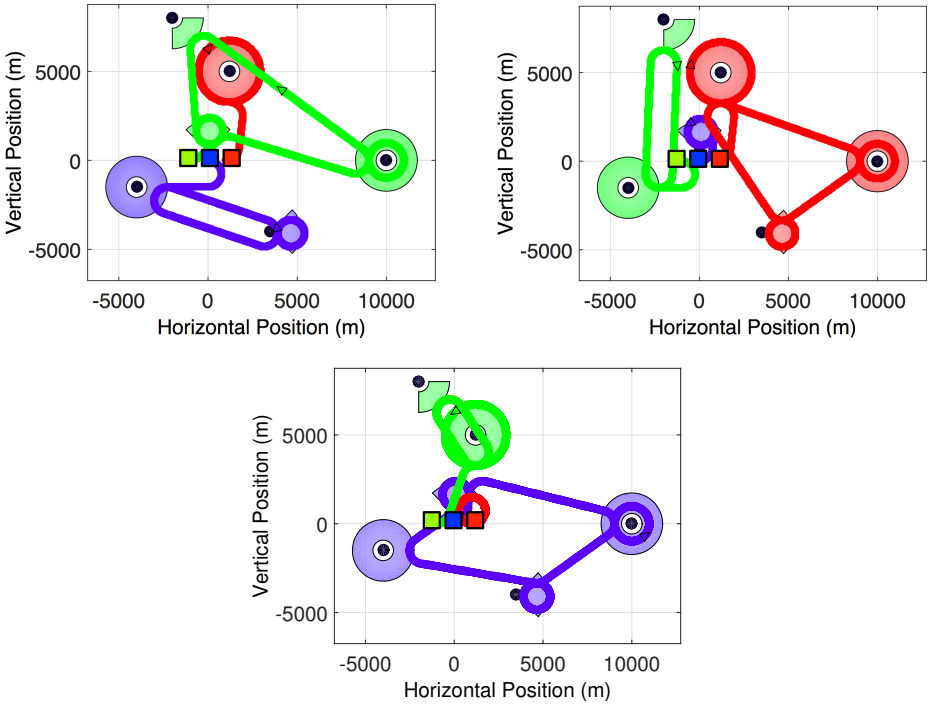

Figure 5.9: Illustration of solutions produced using the decomposition-based solution strategy of Algorithm 15 when target assignment is performed using the "Greedy" strategy (top left), the "Closest" strategy (top right), and the "Random" strategy(bottom).

maximum produced cost of 1702 s; "Closest": all trials produced a cost of 2645 s).

It is clear from Figure 5.9 that the chosen target assignment strategy has a drastic effect on the resultant UAV routes. At first glance, it may seem that the solution produced by the "Greedy" strategy is somewhat counterintuitive, and possibly even inferior to the solution produced by the "Closest" strategy. However, it is important to recall that (i) the UAV is required to make multiple "loops" at some of the targets, (ii) the "Greedy" algorithm is designed to address two conflicting performance metrics simultaneously, namely, the closed trajectory time and the initial maneuver length, and (iii) due to these conflicting performance metrics and the heuristic nature of the algorithm, the "Greedy" algorithm is not guaranteed to provide the global optimum with respect to either of the performance metrics, but simply to provide a reasonable balance between

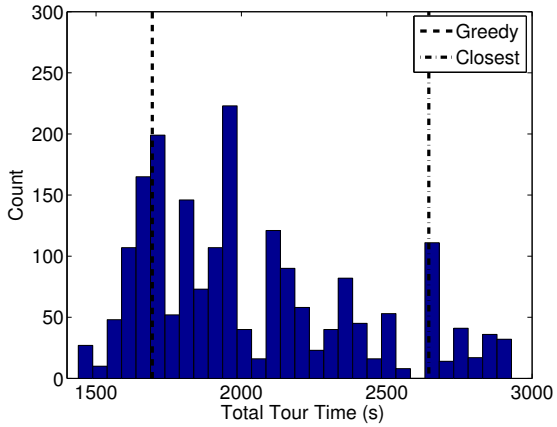

Figure 5.10: Maximum individual UAV tour times (i.e., the value of the objective function in (5.3)), computed over 2000 simulation runs of each implementation of Algorithm 15 for an example surveillance mission.

them. The global time at which a UAV reaches the first target is minimized with the "Greedy" strategy. Further, from Figure 5.10, we see that although, on average, the "Greedy" strategy does not produce the global minimum with respect to the maximum individual UAV tour length, it does produce a tour that has reasonable length. Indeed, even though the mean is about 17.7% longer than the shortest tour produced using random assignment, the mean tour produced using the "Greedy" strategy is shorter than about 82% of the tours produced using the "Random" strategy, and is about 36% shorter than tours produced using the "Closest" strategy. Thus, the "Greedy" strategy provides a reasonable balance between the performance objectives. Of course, the meaning of "reasonable" is dependent upon the application, the system designer, and the end-user. Notice also that, in some instances of Problem 4, the "Closest" and "Random" strategies may not guarantee the feasibility of the produced solutions, while the "Greedy" strategy as implemented here does guarantee feasibility (Lemma 3).

5.6 Chapter Summary

This chapter explored a novel algorithmic framework for constructing unmanned aerial vehicle trajectories for a visibility-constrained, persistent surveillance mission involving static targets. By adopting reasonable constraints on optimal dwell-time behavior and UAV maneuvers, the framework works to balance mission goals, namely, the closed trajectory and the initial maneuver times, while simultaneously accommodating both viewing and dwell-time constraints. In particular, an ϵ-constraint scalarization method is applied to rigorously pose the multi-objective problem as a constrained optimization, which is then approximated by a finite, path-planning problem that results from careful sampling of the UAV configuration space. Solutions to related GTSP problems can then be utilized to construct solutions to the discrete approximation which, in many cases, map directly to globally optimal solutions of the discrete problem. These solutions were then mapped to solutions of the continuous problem. It was shown that, under certain conditions, the complete heuristic procedure is resolution complete in the sense that solution quality generally increases with increasingly fine sampling. A brief expository discussion was also presented to illustrate how the proposed single-vehicle solutions can be paired with target assignment strategies to address certain multi-vehicle extensions in a decomposition-based framework.

Avenues of future research include the expansion to the multi-vehicle case, explicit comparisons with other routing schemes (e.g., Markov chain-based schemes), and an investigation of alternative discretization strategies. In addition, incorporation of uncertain dwell-times and the explicit pairing with other facets of complex missions, e.g. operator analysis of imagery, should be explored.

Part III

Joint Methods

Chapter 6

Joint Scheduling and Routing for Supervisory Surveillance Missions

As discussed in Chapter 1, coordination strategies for supervisory mobile sensor systems must consider several factors. From a human factors perspective, the coordination scheme should help create a high-performance work environment for the operator. In particular, for human processing of sensory data, smart strategies should be employed to ensure that (i) the required tasks are completed, (ii) the operator's cognitive state remains in a high-performance regime, and (iii) performance is robust to behavioral uncertainties. This is the primary goal of the operator-focused methods discussed in Part I. From a robotics perspective, mobile sensor behavior must be coordinated to accomplish mission objectives within an environment that may be large and time-varying. This is the primary goal of the sensor-focused methods discussed in Part II. However, behaviors should also be planned so that (i) tasks requiring operator attention are generated so as to not create bottlenecks, and (ii) uncertainty does not cause undesirable configurations.

While these operator and sensor-focused methods each individually have their own merit, in large part, they do not explicitly take into account the inherent coupling be-

tween the operator and sensor behavior. As such, the full benefit of the human-centered autonomous sensor system may still go unrealized, even if each individual system component is optimized on its own. This chapter seeks to alleviate this issue by jointly optimizing both the operator and sensor behavior within a single coordination framework. In particular, we illustrate this joint optimization approach within the context of a discrete surveillance mission involving a single operator, a team of UAVs, and a set of static ground targets. We note that the problem considered here is derived from the discrete surveillance problem of Chapter 5, together with the scheduling problem of Chapter 3, and the proposed solution framework can, in some sense, be considered as a combination of the methodologies of those two chapters.

6.1 Problem Formulation

6.1.1 Mission Overview and Solution Approach

A team of UAVs, each equipped with a gimbaled, on-board camera, is tasked with collecting surveillance imagery of a set of static targets with known locations. The targets are distributed over a large planar area, so that the UAVs must move within the environment in order to collect complete sensory data. There are no restrictions on which UAV images any particular target, and no single UAV can simultaneously image more than one target. When a UAV reaches a target, it loiters in place (see Section 6.1.3) while simultaneously transmitting its camera feed to a remotely located operator, who takes some amount of time to process the image. The mission is complete when the operator has processed each target exactly once.

We seek a framework to simultaneously generate (i) UAV routes to visit/image the targets, and (ii) the operator's task-processing schedule, i.e., the time at which the oper-

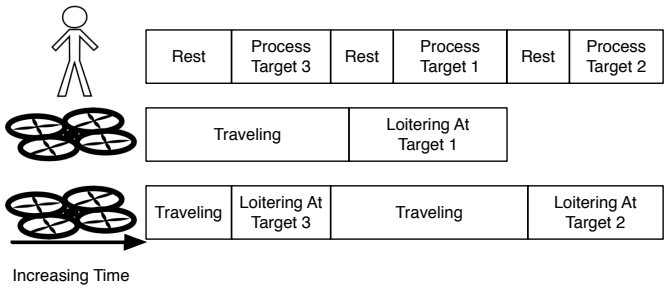

Figure 6.1: Illustration of the coupling between UAV and operator behavior. Notice that the operator cannot start a task until the corresponding UAV reaches the appropriate target, and that the UAV cannot leave the target until the operator finishes processing the task.

ator processes each task. Since target imagery is transmitted in real-time, the availability of target imagery (operator tasks) is determined by the time that the UAVs arrive at each target and, conversely, the required UAV dwell-time at each target is determined by the operator processing time. A diagram illustrating the coupling between the vehicle and operator schedules is shown in Figure 6.1. This relationship introduces a set of constraints to govern the synchronization of human and robotic resources. In addition to satisfying synchronization constraints, the ideal routes/schedule would be such that (i) the operator's *task load* stays within a high-performance regime, and (ii) the time that the UAVs spend loitering unnecessarily, i.e., when the operator is not analyzing their video feed, is minimized. The following subsections expand the mission setup mathematically.

6.1.2 Human Operator Specifications

A single human operator sequentially processes the visual sensory data collected by the UAVs. Assume that all image analysis tasks are of equal importance, and that no two tasks can be executed simultaneously. Further assume that, once a task is started, it must be completed before another task is initiated. As in Chapter 3, we are interested in moderating the operator's *task load*, under the assumption that task load is directly correlated with operator stress and thus, by the typical interpretation of the Yerkes-

Dodson law [76], moderate task-load levels lead to the best performance [18, 76].

We use the same trend-based task-load model as was used in Chapter 3: We represent the operator's task load level as a scalar variable, which, ideally, should be maintained within the finite interval $[\underline{W}, \overline{W}]$. The interval bounds are chosen *a priori*; however, they are treated as "soft" constraints in subsequent optimizations, and thus high precision is not generally required. The operator's task load level evolves as follows (similar to [137, 135]): when the operator is busy, i.e., working on a task, the imposed task load increases by some (task-dependent) amount, and when the operator is idle, the imposed task load decreases by some amount. In this chapter, we assume that task load decrements linearly during idle time with a fixed rate $\delta^- \in \mathbb{R}_{>0}$, e.g., if the operator is idle for time t, then the task load decrement is $\delta^- t$. Task load increments are discussed further in Section 6.1.4.

6.1.3 UAV Specifications

Suppose there are $N \in \mathbb{N}$ UAVs, each responsible for visiting a subset of the targets and transmitting real-time video data to the operator. A UAV must loiter at the target location while the operator processes the associated task. We do not assume a particular dynamic model, although we assume that the time required for traversal of the optimal path between any two UAV configurations can be quantified using a known time-invariant function, i.e., the travel time between two fixed configurations is fixed and known. We collect each UAV's initial configuration in a set V_0 (having size N: create copies if two UAVs share a common initial configuration).

We develop the framework of this chapter under the simplifying assumptions that (i) the UAVs are homogenous, i.e., have identical dynamic and sensing capabilities, and (ii) each UAV is able to "hover" in place. However, some comments are in order: First, heterogenous vehicles can be accommodated via straightforward extensions of the presented

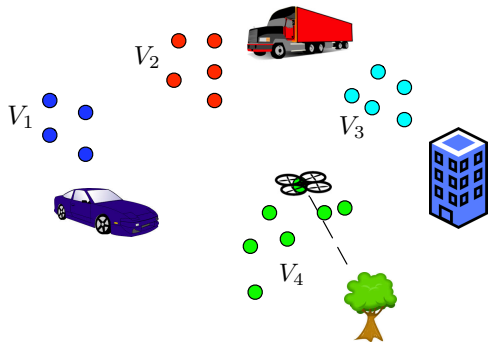

Figure 6.2: Illustration of the sets V_j associated with each target j

methods. Second, the "hovering" assumption is not necessary when using the dynamic strategy of Sections 6.3 and 6.4. As such, the strategies of these sections can also be applied to fixed-wing UAVs, as we demonstrate in Section 6.5.

6.1.4 Target Specifications

A set of M targets must be imaged and analyzed. Associate each target T_j with a finite set V_j of configurations from which the UAV is able to provide the required imagery. That is, to image target j, the UAV must travel to one of the configurations in V_j and hover until the operator processing is complete (see Figure 6.2). Each target T_j can equivalently be considered as an image analysis task, which takes the operator time $\tau_j \in \mathbb{R}_{\geq 0}$ to complete. Initially, assume that τ_j is fixed and known *a priori* for each target (this is relaxed in Section 6.4). As in Chapter 3, let $\Delta W_j \in \mathbb{R}_{\geq 0}$, represent task load increment that results from the operator working on task T_j for the time τ_j.

Remark 22 (Configuration Clusters) *The discrete "clusters"* $(V_1, V_2, \ldots, V_M)$ *arise naturally in sampling-based approximations to continuous planning problems, e.g.,[124]. In our case, these clusters can be thought of as resulting from a discretization procedure similar to that of Section 5.2.1.*

6.1.5 Objectives and Performance Metrics

The goal is to develop an algorithmic framework that simultaneously manages both human and autonomous resources. We consider two performance metrics: (i) the maximum amount (absolute value) by which the operator's task load bounds are violated (both upper and lower) during the mission, and (ii) the aggregate time that the UAVs spend loitering unnecessarily, i.e., loitering at a target before the operator begins processing the video feed. The first metric is considered due to the relation between task load and performance mentioned in Section 6.1.2. The second metric is considered since it is often undesirable to have UAVs spend excessive time loitering at a target, particularly when the target is hostile, e.g., in many military reconnaissance operations [155]. Indeed, unnecessary loitering increases target awareness through increased noise signature, etc. We wish to minimize these metrics by jointly optimizing over (i) the operator schedule, and (ii) the UAV routes.

Using a linear *scalarization* of the multi-objective problem, the full, continuous joint optimization problem is described mathematically as follows.

Problem 5 (Joint Human/UAV Optimization) *Determine both an operator schedule, which dictates when the operator should process each task (target image), and a set of UAV target visitation routes, that together minimize the metric*

$$\begin{aligned} \textit{Mission Cost} = {} & p_\beta(\textit{Max upper task load bound violation}) \\ & + p_\gamma(\textit{Max lower task load bound violation}) \\ & + p_\lambda(\textit{Total unnecessary loiter time}) \end{aligned}$$

where p_β, p_γ, $p_\lambda > 0$ *are fixed parameters.*

The remainder of our analysis is focused on the development of methods for constructing practical solutions to Problem 5. In particular, under a few additional constraints on

UAV behavior, Problem 5 is equivalently represented as a mixed-integer program (MIP), which can be solved using discrete programming methods. In addition, in Section 6.4, we show how these heuristic methods allow the introduction of straightforward, scenario-based techniques for mitigating operator uncertainty.

6.2 Mixed-Integer Programming Formulation

This section develops a MIP approximation to Problem 5. This formulation is based on the following assumptions: (i) each UAV departs its initial location immediately, and departs each successive target viewpoint in its route immediately after the operator completes the associated analysis task, (ii) the time that any UAV takes to travel from a given starting configuration to a given ending configuration is fixed and known, and (iii) travel times satisfy a triangle inequality.

Let V_0 be the set containing the initial configuration of each UAV. We also define a variable K to represent the number tasks in the operator schedule that results from the optimization problem. In this section, we consider the full scheduling operation and thus set $K \coloneqq M$.

6.2.1 Graph Construction

Under the aforementioned assumptions, the problem of finding appropriate UAV target visitation orders reduces to a path-finding problem over a graph. Define the complete, weighted, directed graph $G \coloneqq (V, E, W)$, where $V \coloneqq V_0 \cup V_1 \cup \cdots \cup V_M$ is the union of the clusters $V_0, V_1, \ldots, V_M$ defined in Sections 6.1.3 and 6.1.4, and the weight $W(i, j)$ of edge (i, j) equals the travel time from node i to node j. Assume G also contains zero-weight self-loops, i.e., $(i, i) \in E$ and $W(i, i) \coloneqq 0$ for all $i \in V$. For each UAV n, a valid target visitation order is specified by a path that starts at its initial configuration (contained in

the set V_0), and visits at most one node from any of the clusters $V_1, \ldots, V_M$. Aggregating individual routes, exactly one node from each cluster should be visited by some UAV.

Remark 23 (GTSP) *The UAV route-finding problem is similar to a* multi-vehicle, open, GTSP. *Indeed, we seek paths on G so that exactly one configuration associated to each target is visited by some UAV, and the UAVs need not return to their initial depot. However, the problem of interest herein differs from typical GTSP instances due to operator considerations: the performance metrics considered depend jointly on vehicle behavior and operator behavior, which is not fixed* a priori.

6.2.2 Decision Variables

Let $x_{i,j,k} \in \{0, 1\}$ be a binary decision variable that equals 1 if and only if $(i, j) \in E$ appears in some UAV's route, and the target associated with node j represents the k-th task in the operator schedule (Figure 6.3). The remaining decision variables are continuous: Let $A_k \in \mathbb{R}_{\geq 0}$ be the time that some UAV arrives at the target representing the k-th task in the operator schedule, and let $B_k, C_k \in \mathbb{R}_{\geq 0}$ be the time that the operator begins and completes the k-th task, resp. Next, let $\underline{W}_k, \overline{W}_k \in \mathbb{R}$ represent the operator's task load level immediately before and after processing the k-th task. Note that the subscript k indicates the order in which the operator processes tasks, and not the order in which any single vehicle visits its assigned targets. Finally, define $\beta, \gamma, \lambda \in \mathbb{R}_{\geq 0}$ to act as surrogates to each performance metric: max upper and lower task load bound violation (absolute value), and total unnecessary loiter time. Table 6.1 summarizes the decision variables.

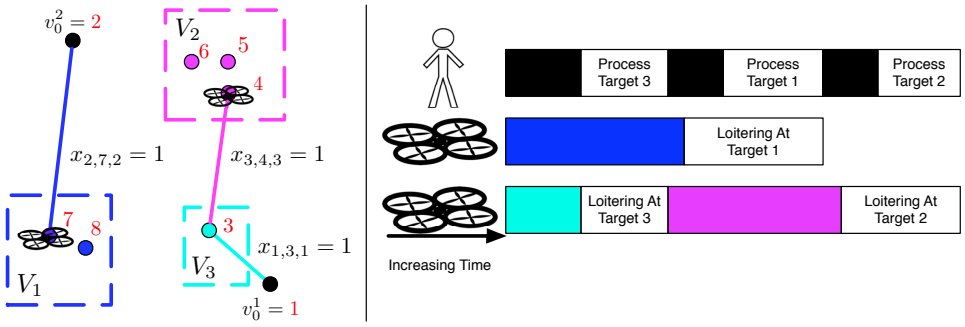

Figure 6.3: Relation between binary decision variables and resulting solution. Notice that $x_{i,j,k} = 1$ if and only if the edge $(i, j) \in E$ is included in some UAV's tour, and the target associated with node j represents the k-th task in the operator's schedule.

Table 6.1: Decision Variables

Variable	Index Set	Description
$x_{i,j,k} \in \{0, 1\}$	$(i, j) \in E$ $k \in \{1, \ldots, K\}$	Indicates if: edge (i, j) is contained in some vehicle's tour and the target associated with j is the k-th operator task
$A_k \in \mathbb{R}_{\geq 0}$	$k \in \{1, \ldots, K\}$	The time that a UAV arrives at the target representing the k-th operator task
$B_k, C_k \in \mathbb{R}_{\geq 0}$	$k \in \{1, \ldots, K\}$	The time the operator begins and completes the k-th scheduled task, resp.
$\underline{W}_k, \overline{W}_k \in \mathbb{R}_{\geq 0}$	$k \in \{1, \ldots, K\}$	The operator's task load level immediately before and after competing the k-th scheduled task, resp.
$\beta, \gamma, \lambda \in \mathbb{R}_{\geq 0}$	-	Variables quantifying performance metrics

6.2.3 Constraints

UAV Path Constraints: The first constraints ensure that a valid UAV tour can be extracted from the decision vector.

$$\sum_{i\in V}\sum_{j\in V_m}\sum_{k=1}^{K} x_{i,j,k} = 1 \qquad \forall m \in \{1,\dots,M\} \tag{6.1}$$

$$\sum_{i\in V_m}\sum_{j\in V}\sum_{k=1}^{K} x_{i,j,k} \leq 1 \qquad \forall m \in \{1,\dots,M\} \tag{6.2}$$

$$\sum_{i\in V}\sum_{j\in V_0}\sum_{k=1}^{K} x_{i,j,k} = 0 \tag{6.3}$$

$$\sum_{i\in V}\left(x_{j,i,k} - \sum_{\hat{k}=1}^{k-1} x_{i,j,\hat{k}}\right) \leq 0 \qquad j \notin V_0, k \in \{1,\dots,K\} \tag{6.4}$$

$$\sum_{k=1}^{K}\sum_{j\in V} x_{i,j,k} \leq 1 \qquad \forall i \in V_0 \tag{6.5}$$

$$\sum_{(i,j)\in E} x_{i,j,k} = 1 \qquad \forall k \in \{1,\dots,K\} \tag{6.6}$$

Equation (6.1) ensures that exactly one node from each cluster is visited, and Equation (6.2) ensures that at most one edge leaves any cluster. Equation (6.3) ensures that vehicles do not return to the initial location set once they leave (we seek *open* UAV paths). Equation (6.4) says that any vehicle leaving a node (excluding the initial node) must enter the same node at an earlier time (let $\sum_{\hat{k}=1}^{0} x_{j,i,\hat{k}} \coloneqq 0$). This constraint also ensures that the first task in the operator schedule coincides with the first target that some UAV visits, and that each UAV's route must begin at its initial node. Equation (6.5) ensures that there is at most one edge leaving each initial configuration, and Equation (6.6) ensures that a single target is chosen for each "slot" in the operator schedule.

UAV Arrival Time Constraint: The following governs the UAV arrival times.

$$\sum_{(i,j)\in E} x_{i,j,k} \left(W(i,j) + \sum_{\hat{\jmath}\in V} \sum_{\hat{k}=1}^{k-1} x_{\hat{\jmath},i,\hat{k}} C_{\hat{k}} \right) = A_k \qquad \forall k \in \{1,\ldots,K\} \tag{6.7}$$

Equation (6.7) says that a UAV's arrival time at a target must equal the time that the operator completes the UAV's previously generated task plus the required travel time.

Task Processing Constraints: The following constraints govern the operator schedule induced by the decision vector.

$$B_k + \sum_{(i,j)\in E} x_{i,j,k}\tau_j = C_k \qquad \forall k \in \{1,\ldots,K\} \tag{6.8}$$

$$A_k \leq B_k \qquad \forall k \in \{1,\ldots,K\} \tag{6.9}$$

$$C_{k-1} \leq B_k \qquad \forall k \in \{2,\ldots,K\} \tag{6.10}$$

Equation (6.8) relates the completion time of a task to the time that processing begins. Equations (6.9) and (6.10) indicate that the operator cannot start a task until (i) a UAV arrives at the target, and (ii) the previous task is complete.

Task Load Evolution Constraints: The following constraints govern the evolution of the operator's task load during the mission.

$$\underline{W}_k + \sum_{(i,j)\in E} x_{i,j,k}\Delta W_j = \overline{W}_k \qquad \forall k \in \{1,\ldots,K\} \tag{6.11}$$

$$W_0 - \delta^- B_1 = \underline{W}_1 \tag{6.12}$$

$$\overline{W}_{k-1} - \delta^-(B_k - C_{k-1}) = \underline{W}_k \qquad \forall k \in \{2,\ldots,K\} \tag{6.13}$$

Equation (6.11) ensures that workload increments are allocated properly while Equations (6.12) and (6.13) ensure that workload decrements are defined allocated properly.

Performance Constraints: The final constraints quantify performance metrics.

$$\overline{W}_k - \overline{W} \leq \beta \qquad \forall k \in \{1, \ldots, K\} \tag{6.14}$$

$$\underline{W} - \underline{W}_k \leq \gamma \qquad \forall k \in \{1, \ldots, K\} \tag{6.15}$$

$$\sum_{k=1}^{K} B_k - A_k = \lambda \tag{6.16}$$

6.2.4 MIP Formulation

A mixed-integer nonlinear program (MINLP) approximation of Problem 5 is defined in Problem 6.

Problem 6 (Scheduling/Routing MINLP) *Determine values for each of the decision variables (Table 6.1) that are optimal with respect to the following problem:*

$$\begin{aligned} &\textit{Minimize:} && p_\beta \beta + p_\gamma \gamma + p_\lambda \lambda \\ &\textit{Subject To:} && \textit{Constraints } (6.1) - \ (6.16), \end{aligned} \tag{6.17}$$

where $K \coloneqq M$*, and* p_β*,* p_γ*,* $p_\lambda > 0$ *are constant parameters.*

Notice that the only nonlinearity in (6.17) is due to (6.7). In general, the non-linearity can be eliminated by augmenting the problem with a set of auxiliary variables and constraints, although this linearization results in significantly larger problems in general. However, when $N = 1$, additional problem structure emerges that allows for elimination of the non-linearity without increasing the problem size. These results are formalized here.

Theorem 8 (Linearization) *The non-linear program* (6.17) *is equivalent to a MILP.*

Moreover, in the single vehicle case ($N = 1$), an equivalent MILP exists with the same number of binary decision variables, continuous decision variables, and algebraic constraints as its non-linear counterpart (6.17).

Proof: Let $I \coloneqq \{(i,j,k,\hat{\jmath},\hat{k}) \mid i,j,\hat{\jmath} \in V, k \in \{2,\dots,M\}, \hat{k} \in \{1,\dots,k-1\}\}$, and define decision variables $y_{i,j,k,\hat{\jmath},\hat{k}} \in \{0,1\}$, $D_{i,j,k,\hat{\jmath},\hat{k}} \in \mathbb{R}_{\geq 0}$ for each $(i,j,k,\hat{\jmath},\hat{k}) \in I$. Notice $|I| = |V|^3 \sum_{k=2}^{M}(k-1) = \frac{1}{2}(M(M-1))|V|^3$. Let $\widehat{C} > 0$ be a very large constant and define constraints:

$$\sum_{i\in V}\sum_{j\in V} x_{i,j,k} W(i,j) + \sum_{i\in V}\sum_{j\in V}\sum_{\hat{\jmath}\in V}\sum_{\hat{k}=1}^{k-1} D_{i,j,k,\hat{\jmath},\hat{k}} = A_k \qquad \forall k \in \{2,\dots,K\} \tag{6.18}$$

$$y_{i,j,k,\hat{\jmath},\hat{k}} \leq x_{i,j,k} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.19}$$

$$y_{i,j,k,\hat{\jmath},\hat{k}} \leq x_{\hat{\jmath},i,\hat{k}} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.20}$$

$$x_{\hat{\jmath},i,\hat{k}} + x_{i,j,k} - 1 \leq y_{i,j,k,\hat{\jmath},\hat{k}} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.21}$$

$$D_{i,j,k,\hat{\jmath},\hat{k}} \leq \widehat{C} y_{i,j,k,\hat{\jmath},\hat{k}} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.22}$$

$$D_{i,j,k,\hat{\jmath},\hat{k}} \leq C_{\hat{k}} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.23}$$

$$C_{\hat{k}} - \widehat{C}(1 - y_{i,j,k,\hat{\jmath},\hat{k}}) \leq D_{i,j,k,\hat{\jmath},\hat{k}} \qquad \forall (i,j,k,\hat{\jmath},\hat{k}) \in I \tag{6.24}$$

This linear constraint set is equivalent to (6.7) for $\widehat{C}$ sufficiently large: under (6.19) - (6.24), we have $D_{i,j,k,\hat{\jmath},\hat{k}} = y_{i,j,k,\hat{\jmath},\hat{k}} C_{\hat{k}} = x_{i,j,k} x_{\hat{\jmath},i,\hat{k}} C_{\hat{k}}$; thus, (6.18) is equivalent to (6.7). Thus, an equivalent MILP results from replacing (6.7) with (6.18) - (6.24) in (6.17).

If $N = 1$, the operator's task processing order is identical to the UAV's target visitation order. As such, the arrival time A_k equals the time that the operator finishes the $k-1$-st task, plus the required UAV travel time to the next target. Formally, when

$N = 1$, Equation (6.7) is equivalent to

$$\sum_{i \in V} \sum_{j \in V} x_{i,j,1} W(i,j) = A_1$$
$$C_{k-1} + \sum_{i \in V} \sum_{j \in V} x_{i,j,k} W(i,j) = A_k \qquad \forall k \in \{2, \ldots, K\}.$$

Replacing (6.7) with this linear alternative in (6.17), results in a MILP. ■

The proof of Theorem 8 also provides a better understanding of the size of the equivalent MILPs. For instance, notice that, since the number of vehicles N is typically small, the problem size is typically dominated by the number of targets M and the size of the graph G (which depends on M and the number of viewpoints associated with each target). If each target has $P \in \mathbb{N}$ associated viewpoints, i.e., $|V_j| = P$ for all $j \in \{1, \ldots, M\}$, and the number of vehicles N is fixed, then, as M and P jointly tend to infinity, the number of binary decision variables, continuous decision variables, and algebraic constraints in the MINLP (6.17) are $O(M^3P^2)$, $O(M)$, and $O(M^2P)$. When $N = 1$, an equivalent MILP of the same size exists; however, by the proof of Theorem 8, we see that for general, multi-vehicle missions, the number of binary decision variables, continuous decision variables, and algebraic constraints required in an equivalent MILP are each $O(M^5P^3)$. Due to the availability of high-quality MILP solvers, such as GLPK [149], CPLEX [156], and MATLAB's `INTLINPROG` solver [157], Theorem 8 may evoke a practical strategy for some missions. However, even when $N = 1$, problems may become large and computationally complex. In response, the following section develops a dynamic, heuristic strategy for constructing solutions to general instances of Problem 6.

6.3 Dynamic Solution Strategy

This section proposes a dynamic framework to construct solutions to Problem 6. Here, each time the operator finishes a task, a comparatively small-scale MILP is solved to select the next UAV destinations and the impending portion of the operator schedule. In particular, each re-planning operation results in a partial operator schedule containing at most N tasks, the first of which is the next task to be executed. In this section, we let $K := \min\{N, M\}$, where M is understood here as the number of targets that have not yet been processed when a re-plan operation is initiated.

The MILP governing each re-plan operation depends on the UAV statuses. Consider three status classifications: (i) `AVAILABLE`: the UAV has not been assigned any targets or the operator has just finished processing the UAV's imagery, (ii) `ARRIVED`: the UAV has reached its destination and is awaiting the operator's attention, and (iii) `TRANSIT`: the UAV is en route to its next destination. Assume that the status of each UAV is available to the optimizer during any re-planning operation. Further, the re-planning operation is performed under the following behavioral assumptions: (i) if its status is `ARRIVED`, then the UAV should remain loitering until the operator processes its task, and (ii) if its status is `TRANSIT`, then the UAV should continue on to its destination, i.e., the UAV should not be re-routed. These are natural assumptions that serve the dual purpose of both reducing computation and preventing excessive changes to UAV routes that may be undesirable from an operator or mission-planning standpoint[1]. We note, however, that these assumptions can be relaxed through straightforward manipulations to the proposed methods (Remark 24).

Broadly, the proposed dynamic solution strategy uses the following procedure:

1. Initialize each UAV status as `AVAILABLE`,

[1]These assumptions also serve to prevent "switching" behavior that may emerge in dynamic extensions in which new targets arrive dynamically, although this case is not considered here.

2. Formulate and solve the re-planning MILP,

3. Direct the UAVs for to their first destination, and instruct the operator to execute the first task in the resulting schedule,

4. When the first task is complete, reformulate and solve the re-planning MILP, and

5. Repeat steps 3 and 4 until all tasks are complete.

The remainder of this section details the formulation and solution of the re-planning MILP described in step 2 and 4.

6.3.1 Graph Modifications

Consider some instant at which the re-planning operation is to be executed, and assume, without loss of generality, that $V_1, V_2, \ldots, V_M$ represent the node clusters associated with the targets that have not yet been processed (we retain this assumption for the remainder of this section). We reconstruct the graph G as follows: First, re-define $V_0 \coloneqq \{i_1, i_2, \ldots, i_N\}$, where i_n represents UAV n's current configuration. Second, for each $n \in \{1, \ldots, N\}$, define a set V_n^{Dest} consisting of nodes to which UAV n can travel prior to another re-assignment. That is, define each set V_n^{Dest} as follows:

1. if UAV n has status `AVAILABLE`, then $V_n^{\text{Dest}} \coloneqq V_1 \cup \cdots \cup V_{\widehat{M}}$, where we have assumed, without loss of generality, that $\{V_1, V_2, \cdots, V_{\widehat{M}}\}$ collects all node clusters that have not yet been visited by a UAV and are not the destination of any UAV with status `TRANSIT`, and

2. if UAV n has status `ARRIVED`, then $V_n^{\text{Dest}} \coloneqq \{i_n\}$, where i_n is UAV n's current configuration, and

3. if UAV n has status `TRANSIT`, then $V_n^{\text{Dest}} := \{j_n\}$, where $j_n \in V$ is UAV n's current destination.

Redefine $V := V_0 \cup \left(\bigcup_{n=1}^{N} V_n^{\text{Dest}}\right)$, and redefine the edge set $E := \{(i_n, j) \mid i_n \in V_0, j \in V_n^{\text{Dest}}\}$. Finally, define the weight operator W to be consistent with the new edges.

The remaining sections formulate the re-planning MILP using the modified graph G.

Remark 24 (Re-Routing) *The set V_n^{Dest} contains a single element whenever UAV n has status `ARRIVED` or `TRANSIT`. As such, UAV n will not be directed to a new destination by the re-planning operation (see (6.27)). Alternatively, one could allow re-routing by instead defining each V_n^{Dest} to contain all nodes that are associated with some target that has not yet been processed, regardless of UAV status.*

6.3.2 Optimization Variables

The decision variables are defined identically to those of section 6.2.2. However, there are two subtle differences: First, the value of the index k refers to the post-replan processing order, rather than the global processing order as before. Second, for re-planning, we only require a variable $x_{i,j,k} \in \{0, 1\}$ for each index triplet in the set $\{(i, j, k) \,|\, i = i_n, j \in V_n^{\text{Dest}}, k \in \{1, \ldots, K\}, n \in \{1, \ldots, N\}\}$. Notice also that, for dynamic re-planning, the value of k corresponds to the post-replan processing order. This restriction typically produces a significantly smaller problem in comparison to (6.17). For instance, when N is fixed and $|V_j| = P$ for all $j \in \{1, \ldots, M\}$, then, as M and P jointly tend to infinity, the number of binary decision variables, continuous decision variables, and algebraic constraints are $O(MP)$, $O(M)$, and $O(M)$, resp.

6.3.3 Constraints

UAV Path Constraints: Define $\mathcal{N}$ as the set of UAV indices with status `AVAILABLE`, and consider the following constraints.

$$\sum_{n\in\mathcal{N}}\sum_{j\in V_m}\sum_{k=1}^{K} x_{i_n,j,k} \leq 1 \qquad \forall m \in \{1,\ldots,\widehat{M}\} \tag{6.25}$$

$$\sum_{j\in V_n^{\text{Dest}}}\sum_{k=1}^{K} x_{i_n,j,k} \leq 1 \qquad \forall n \in \mathcal{N} \tag{6.26}$$

$$\sum_{j\in V_n^{\text{Dest}}}\sum_{k=1}^{K} x_{i_n,j,k} = 1 \qquad \forall n \notin \mathcal{N} \tag{6.27}$$

$$\sum_{(i,j)\in E} x_{i,j,k} = 1 \qquad \forall k \in \{1,\ldots,K\} \tag{6.28}$$

Constraint (6.25) says that at most one node from any cluster can be visited. Constraint (6.26) says that any `AVAILABLE` UAV is assigned at most one new destination, while constraint (6.27) says that UAVs with status `WAITING` or `TRANSIT` do not get re-routed. Constraint (6.28) ensures that the maximum number of UAVs are assigned a destination, i.e., if at least N targets have not been processed, then N UAVs are assigned a destination; otherwise, all remaining targets are assigned a UAV.

Remark 25 (Running Cost) *Constraint* (6.28) *ensures that a meaningful solution is produced by the re-planning MILP: Without* (6.28)*, UAVs may remain unassigned when there are still unprocessed targets, in which case the running cost is not a meaningful predictor of the overall mission cost. For example, if all UAVs are* `AVAILABLE`*, then, without* (6.28)*, the zero vector is an optimal choice for the binary variables, making the running cost predicted by the re-planning MILP equal to zero.*

UAV Arrival Time Constraints: The following govern the UAV arrival times:

$$\sum_{n=1}^{N} \sum_{j \in V_n^{\text{Dest}}} x_{i_n,j,k} W(i_n, j) = A_k \qquad \forall k \in \{1, \ldots, K\}. \tag{6.29}$$

Recall that, if UAV n has status `ARRIVED`, then $V_n^{\text{Dest}} \coloneqq \{i_n\}$. Since $W(i_n, i_n) = 0$, Equation (6.29) implies that the arrival time associated a UAV having status `ARRIVED` is zero. As such, the operator is permitted to begin processing any task associated with such a UAV immediately. In contrast, if a UAV has status `TRANSIT` or `AVAILABLE`, then the weight $W(i_n, j)$ captures the time required for the UAV to travel from its current location, i_n, to its new destination in the set V_n^{Dest}. As such, the operator cannot start any such task until the corresponding UAV arrives at its destination.

Task Processing, Task Load Evolution, and Performance Constraints: As before, we enforce the constraints (6.8)- (6.16), where W_0 is understood as the operator task load level at the re-plan onset. Define $\beta_0, \gamma_0 \in \mathbb{R}_{\geq 0}$ as the maximum upper and lower task load bound violations that have occurred prior to the current re-plan, and introduce two final constraints:

$$\beta \geq \beta_0 \tag{6.30}$$

$$\gamma \geq \gamma_0. \tag{6.31}$$

These constraints ensure that the running cost is correlated with the global mission cost.

6.3.4 MILP Formulation

The MILP governing the re-plan operation is formally expressed as Problem 7.

Problem 7 (Re-planing operation as a MILP) *Determine values for each of the*

decision variables in Table 6.1 that are optimal with respect to the following problem:

$$\begin{aligned} &\textit{Minimize:} && p_\beta \beta + p_\gamma \gamma + p_\lambda \lambda \\ &\textit{Subject To:} && \textit{Constraints } (6.8) - (6.16) \textit{ and } (6.25) - (6.31), \end{aligned} \tag{6.32}$$

where $K \coloneqq \min\{N, M\}$*,* W_0 *is the operator's current task load, and* p_β*,* p_γ*,* $p_\lambda > 0$ *are fixed parameters.*

Remark 26 (Task Processing) *The optimization problem* (6.32) *predicts the incremental cost over the impending portion of the operator's schedule under the implicit assumption that the operator processes one task generated by each UAV before processing any other tasks (see* (6.28)*). However, this assumption does not place an explicit constraint on the global structure of the operator schedule. That is, assuming Problem 7 is solved each time a task is completed, then it is still possible for the operator process two tasks in a row that are generated by the same UAV.*

6.4 Uncertain Processing Times

The analysis presented thus far has assumed that the processing time associated with each target is known *a priori*. Indeed, strictly speaking, known processing times are required to ensure that the optimization framework correctly relates task start times and completion times (see (6.8)), which are used to predict subsequent UAV arrival times. This assumption does not hold in most realistic systems, since operator behavior is subject to various types of uncertainty.

The most common approach to addressing processing time uncertainty is to use a single realization of each uncertain parameter within the optimization framework, e.g.,

use expected values. Sophisticated robust optimization schemes may be useful (see [158]), although these methods generally require bounded uncertainty sets and very particular problem structure. In contrast, *scenario-based* robust optimization schemes use discrete samples as approximations to the uncertainty sets, and require optimization constraints to be satisfied for *all* of the sampled conditions, rather than just one [159, 160]. As a result, solutions produced by scenario-based schemes are likely to be of a high quality for many realizations of uncertain parameters.

This section introduces a straightforward, scenario-based extension, similar to that of Chapter 3, to the framework of Section 6.3, with the goal of achieving robustness to uncertainty in operator processing times. That is, the optimization problem developed here is a robust alternative to (6.32). We use a sampling-based method since (i) the complex, coupled nature of the joint routing/scheduling problem makes it difficult to predict "worst"-case parameter values directly, (ii) typical processing time distributions used to model perceptual decision-making are skewed and have unbounded support [152, 32]; as such, expected values may be inaccurate processing time predictors and methods requiring bounded support would require enforcement of additional constraints, (iii) sampling parameters allow a straightforward means of tuning the "degree of robustness" provided in order to strike a balance with computational complexity, and (iii) scenario-based schemes are simple, intuitive, and use a straightforward procedure that does not require any particular uncertainty distribution.

6.4.1 Constructing Scenarios

Suppose the operator's processing time for each target (task) T_j is realized according to a probability density function $f_j : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. For each j, generate a set of $Q \in \mathbb{N}$ possible processing times $\{\tau_j^1, \tau_j^2, \ldots, \tau_j^Q\}$, where $\tau_j^q \sim f_j$ for $q \in \{1, \ldots, Q\}$. For each

$q \in Q$, the set $\{\tau_1^q, \tau_2^q, \ldots, \tau_M^q\}$ defines a *scenario*. Note that this sampling process associates each target T_j with a set of processing times, rather than a single realization.

6.4.2 Optimization Variables

The scenario-based extension of the MILP (6.32) contains only slightly modified decision variables to accommodate the generated processing time sets. In particular, the binary variables $\{x_{i,j,k} \in \{0,1\}\}$, the arrival time variables $\{A_k\}$, and the performance variables β, γ, λ are identical to those of (6.32). For the remaining decision variables, we introduce duplicates, indexed by the superscript q, that are each associated with one scenario. That is, for each $q \in \{1, \ldots, Q\}$, we define unique decision variables $B_k^q, C_k^q, \overline{W}_k^q, \underline{W}_k^q$ that are associated with the q-th scenario.

Remark 27 (Arrival Times) *Scenario-dependent arrival times A_k are not required, since each re-plan only selects each UAV's next destination. Since UAVs begin moving immediately, the arrival times calculated by the re-plan operation are independent of previous operator processing times.*

6.4.3 Constraints

The decision variables are subject to constraints (6.14), (6.15), and (6.25) - (6.29). The scenario-based analogs of the remaining constraints are written as follows:

$$B_k^q + \sum_{(i,j)\in E} x_{i,j,k}\tau_j^q = C_k^q \qquad \forall k \in \{1,\ldots,K\}, q \in \{1,\ldots,Q\} \tag{6.33}$$

$$A_k \leq B_k^q \qquad \forall k \in \{1,\ldots,K\}, q \in \{1,\ldots,Q\} \tag{6.34}$$

$$C_{k-1}^q \leq B_k^q \qquad \forall k \in \{2,\ldots,K\}, q \in \{1,\ldots,Q\} \tag{6.35}$$

$$\underline{W}_k^q + \sum_{(i,j)\in E} x_{i,j,k}\Delta W_j = \overline{W}_k^q \qquad \forall k \in \{1,\ldots,K\}, q \in \{1,\ldots,Q\} \tag{6.36}$$

$$\overline{W}_k^q - \overline{W} \leq \beta \qquad \forall k \in \{1, \ldots, K\}, q \in \{1, \ldots, Q\} \tag{6.37}$$

$$\underline{W} - \underline{W}_k^q \leq \gamma \qquad \forall k \in \{1, \ldots, K\}, q \in \{1, \ldots, Q\} \tag{6.38}$$

$$\sum_{k=1}^{K} B_k^q - A_k \leq \lambda \qquad \forall q \in \{1, \ldots, Q\} \tag{6.39}$$

$$B_1^q - B_1^{q-1} = 0 \qquad \forall q \in \{2, \ldots, Q\} \tag{6.40}$$

Constraints (6.33) - (6.39) are analogous to those in Section 6.3. Constraint (6.40) ensures that a unique start time is produced for the first task in the operator's new schedule (other start times should remain scenario-dependent). Since a re-plan operation is performed whenever a task is completed, an unambiguous operator schedule can always be extracted from the MILP solution as a result of this setup.

Remark 28 (Task Load Increments) *The scenario-based scheme also allows the possibility of scenario (time)-dependent increments ΔW_j. For example, if $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ captures the relationship between processing time and the resultant task load increment, then for each q, one can define $\{\Delta W_1^q, \Delta W_2^q, \ldots, \Delta W_M^q\}$, where $\Delta W_j^q \coloneqq g(\tau_j^q)$ for each $j \in \{1, \ldots, M\}$. The subsequent robust MILP is formulated by replacing each instance of ΔW_j with ΔW_j^q in the optimization constraints.*

6.4.4 Scenario-based, Robust MILP

The scenario-based extension to Problem 7 is stated as follows.

Problem 8 (Robust Re-planning as an MILP) *Determine values for the decision variables described in Section 6.4.2 that are optimal with respect to the following problem:*

$$\begin{aligned} &\textit{Minimize:} && p_\beta\beta + p_\gamma\gamma + p_\lambda\lambda \\ &\textit{Subject To:} && \textit{Constraints } (6.14), (6.15),\ (6.25) -\ (6.29),\ (6.33) -\ (6.39) \end{aligned} \tag{6.41}$$

where $K \coloneqq \min\{N, M\}$*,* W_0 *is the operator's current task load, and* p_β*,* p_γ*,* $p_\lambda > 0$ *are fixed parameters.*

6.5 Simulation Studies and Discussion

This section contains simulations studies and discussion to illustrate the advantages and limitations of the proposed solution strategies. In all studies, optimal MILP solutions were estimated using the stand-alone `glpsol` solver, which is included in GLPK v. 4.60 [149].

6.5.1 Performance of the Dynamic Solution Strategy

The first study compares the performance of the receding-horizon strategy of Section 6.3 to that of an *a priori* planning method, which constructs a complete solution by solving the MINLP (6.17) at the mission onset. To allow direct computation of solutions to (6.17), we consider a small-scale mission with 2 UAVs (with "hovering" capability) and 4 targets, each with a single viewpoint ($|V_j| = 1$ for all j). The UAVs move with speed 39 m/s, and travel times are defined as the minimal straight-line travel time between configurations. The initial task load is $W_0 = 0.5$, with an increment/decrement rate of 0.001 during busy/idle times, and the operator takes exactly 483.32 s to complete each task ($\tau_j = 483.32$ s $\forall j$). The desired task-load range is $[\underline{W}, \overline{W}] = [0.2, 0.8]$ and $p_\beta = p_\gamma = 1$. In each simulation run, UAV initial locations and target locations are selected randomly

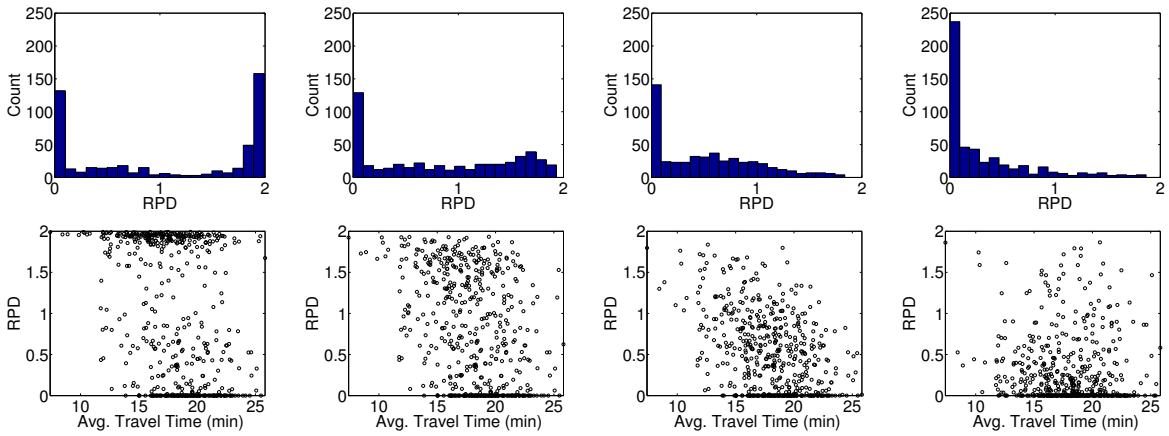

Figure 6.4: Relative percent difference (RPD) between solutions produced using the dynamic routing framework of Section 6.3 and the *a priori* scheme for (left to right) $p_\beta = 0.1, 0.01, 0.001, 0.0001$.

(uniform distribution) within an $80,000 \times 80,000$ m region, and an overall mission cost was calculated under both dynamic re-planning (Section 6.3) and *a priori* planning for each of 4 p_λ values: $0.1, 0.01, 0.001, 0.0001$. Direct solutions to (6.17) were constructed by solving the equivalent MILP (Theorem 8).

Figure 6.4 shows the error generated, for each of 500 simulated missions, between the dynamic and *a priori* strategies in the form of a histogram and as a scatter plot (plotted against the average travel times, i.e., edge weight in the complete graph G), for each of the p_λ conditions. The error is reported as the *relative percent difference* (RPD), defined

$$\text{RPD} = 2\ (\text{dynamic cost} - \textit{a priori}\ \text{cost})/(\text{dynamic cost} + \textit{a priori}\ \text{cost}).$$

Figure 6.4 suggests that the solutions provided by the dynamic heuristic are closest to the optimal solution to (6.17) when: (i) UAV travel times between destinations are large, and (ii) p_λ is small in comparison to p_β, p_γ, i.e., workload considerations are more prominent than unnecessary loitering considerations. Outside of these regimes, the dynamic heuristic performs significantly worse than the *a priori* method. However, a few

comments are in order: First, the very small size of this example allows for accurate estimation of optimal solutions to Problem 6 through transformation to and subsequent solution of an equivalent MILP. In general, the computational burden involved with this and other similar methods will not allow for the direct construction of a high-quality, complete solution to (6.17) in reasonable time. Thus, *a priori* methods are often impractical for general use. Second, instances in which the dynamic heuristic performs poorly are often a result of unnecessary UAV utilization. For example, the dynamic heuristic often assigns the same number of targets to each UAV, which, when targets are close together, forces the UAVs to loiter at target destinations for long time intervals while awaiting operator attention. In contrast, the *a priori* method will avoid these penalties by leaving a subset of the UAVs un-utilized. Therefore, performance of the dynamic heuristic could be improved by simply considering only a subset of the available UAVs when planning. This is illustrated by Figure 6.5, which shows the relative error produced for the $p_\lambda = 0.1$ case when one of the UAVs is omitted, i.e., errors are reported for 500 simulation runs for a single-vehicle mission ($N = 1$) using the same mission setup as before. We note that the analogous plots for the other p_λ conditions are nearly identical and are thus omitted. This discussion suggests that poor performance of the dynamic heuristic is potentially an indicator of a poorly designed mission, i.e., too many UAVs are assigned to the mission. Finally, *a priori* methods that seek to solve Problem 6 directly do not readily extend to cases in which processing times are uncertain or in which UAVs do not have hovering capability, whereas the dynamic scheme easily extends to incorporate these scenarios (Section 6.5.2). A thorough exploration of alternative formulations to improve performance with respect to direct solution methods is left as a topic of future work.

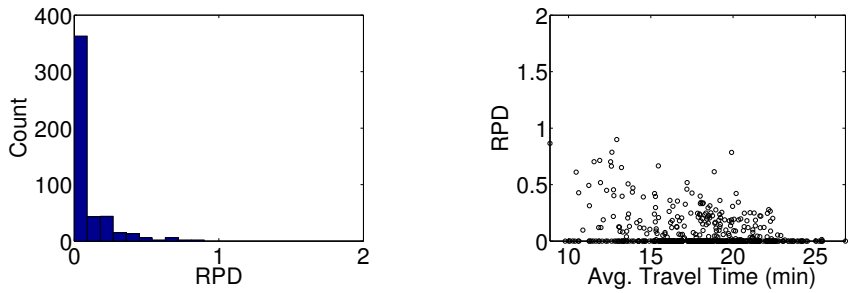

Figure 6.5: RPD between the solutions produced by the dynamic planning scheme of Section 6.3 and the *a priori* planning scheme for a single-vehicle mission with $p_\lambda = 0.1$.

6.5.2 Scenario-Based Planning

The next study demonstrates the utility of the scenario-based scheme of Section 6.4. We consider a realistic example which relaxes many assumptions on UAV and operator behavior that were used in formulating the solution methods discussed in this chapter. Thus, this study not only shows robustness qualities of the scenario-based method, but also illustrates its flexibility in accommodating constraints such as (i) fixed-wing UAVs, which cannot "hover" and have a minimum turning radius, (ii) user-specified imaging constraints, and (iii) uncertain processing times.

Mission Setup: We consider a mission involving $N = 3$ fixed-wing UAVs and $M = 6$ static ground targets. The UAVs are each modeled as a Dubins vehicle, which flies with a fixed forward speed of 39 m/s, a fixed altitude of 1000 m, and a minimum turning radius of 750 m. The initial location and heading of each UAV is $(0, 0)$ and 0, resp. We neglect all other dynamic effects, e.g., drag or wind, and we do not consider the possibility of collision. The mission requires that each target be imaged with a tilt-angle (measured below the horizontal flight plane) within the range $[\pi/8, 3\pi/8]$, similar to the $\texttt{BEH}_j = \texttt{ANY}$ case from the problem discussed in Chapter 5. The location of each target is presented in the table on the left-side of Figure 6.6.

Index, j	Location, t_j
1	$(10000, 0)$ m
2	$(5000, 500)$ m
3	$(1200, 5000)$ m
4	$(-400, -15000)$ m
5	$(3000, -5000)$ m
6	$(-3000, -4000)$ m

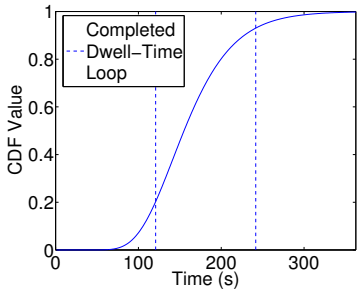

Figure 6.6: Target locations (left) and processing time cumulative distribution function (CDF) (right) for the example surveillance mission

Processing times are modeled as random variables, each realized according to a probability density function $f : \mathbb{R}_{>0} \to \mathbb{R}_{\geq 0}$. More specifically, for each $j \in \{1, \ldots, M\}$, the processing time is log-normally distributed with parameters $\mu = 5.044$, $\sigma = 0.25$ (log mean and log standard deviation). Intuitively, these parameters indicate that, at any single target, the operator will usually ($\sim 75\%$ probability) require the UAV to make between 1 and 2 complete dwell-time loops in order to complete the analysis task. The cumulative distribution function (CDF) generated by f is shown in the right side of Figure 6.6. Let $W_0 = 0.2$, and $[\underline{W}, \overline{W}] = [0.2, 0.8]$, and assume a time-dependent, linear task load evolution model, in which task-load levels increment or decrement by $0.001t$ when the operator is busy or idle for time t, resp. Select $p_\beta = p_\gamma = 10$ and $p_\lambda = 0.01$.

Solution Approach: We apply the dynamic scheduling/routing framework to the example mission using the following steps: First, we construct $V_1, \ldots, V_n$ by applying applying the discretization strategy of Chapter 5; namely, each set V_j consists of discrete points in the configuration space ($\mathbb{R}^2 \times [0, 2\pi)$) from which the UAV is able to immediately begin an appropriate loitering pattern at the target j. More precisely, each sample $(x, \theta) \in V_j \subset \mathbb{R}^2 \times [0, 2\pi)$ has a heading θ that points in a direction tangent, at the location x, to a circle of radius 750 m (the minimum possible turning radius), which lies entirely within the annulus of locations from which the tilt-angle specification is met.

Travel times are defined as the time required for the UAV to traverse the optimal Dubins path between configurations. We then apply the scenario-based scheme of Section 6.4 to dynamically find UAV routes and operator schedules, using scenario-dependent task load increments according to the linear model, i.e., for each sampled processing time τ_j^q, we let $\Delta W_j^q := 0.001\tau_j^q$ (see Remark 28).

Performance Analysis: For comparison, we consider a baseline solution method that ignores task load levels, as well as the need to synchronize operator and robotic resources. In particular, the chosen baseline method constructs UAV routes according to a dynamic implementation of the multi-vehicle method of Section 5.5 (where straightforward modifications have been made to produce *open*, rather than *closed*, routes). Namely, each time the baseline planner is called, complete UAV paths are constructed under an assumed processing time of 362.49 s at each target (the time required for the UAV to complete 3 dwell-time loops). Intuitively, this choice can be thought of as a type of "worst"-case processing time, since the probability of realizing a longer processing time is $< .01$. The baseline solution then assumes that the operator processes tasks as soon as possible, in a first-come first-serve basis. UAVs dwell at each successive target destination until the operator processes the task. Each time the operator finishes a task, the planner is called once again and the assignment/routing process is repeated over the remaining targets. For consistency with the methods of Section 5.5, the baseline method selects ϵ to be as small as possible during initial route construction (see Lemma 5.5), while each successive re-plan selected $\epsilon = \infty$.

We compare the baseline method to the scenario-based method over multiple simulated mission executions. Here, the "actual" processing times (unknown to the planner) were sampled from the distribution f (whose CDF is shown in Figure 6.6). Figure 6.7 shows an example mission progression for the baseline solution (left column) and for the

scenario-based strategy of Section 6.4 (right column), using identical "actual" processing times (processing time realizations) in each case. Figure 6.8 depicts the utilization of UAV and human resources during the same mission. Notice, in particular, how the optimal routing strategy differs when using the joint optimization as opposed to the baseline. Indeed, in the joint optimization case, longer UAV routes between targets may be chosen if the operator's task load level is high or to avoid unnecessary loitering times. In contrast, the baseline solution attempts to minimize route lengths and ignores operator behavior.

Figure 6.9 shows the costs obtained over 100 simulated missions using various planning strategies, in both tabular and graphical form. The strategies included are: (i) the baseline method, (ii) two instances of the dynamic scheme of Section 6.3, one of which assuming expected processing time values (labeled "Expect") and one assuming "worst"-case processing times (as in the baseline case), and (iii) three instances of the scenario-based method of Section 6.4 corresponding to $Q = 1$, 5, and 10. Different "actual" processing times were sampled for each run, but were held constant across conditions, i.e., each simulation run was performed by first sampling f to obtain "actual" processing times, and subsequently testing each condition using the realized values. Any points whose distance from the median is further than 2.5 times the inter-quartile range (IQR) are considered outliers. As expected, all of the joint optimization schemes performed significantly better than the baseline, and larger numbers of scenarios typically reduced the cost variance, indicating increased robustness. Using expected values generally produced higher costs and much higher cost variance than the scenario-based scheme, highlighting the risk involved with this type of naive sampling in the presence uncertainty.

With respect to the "worst"-case approach, we remark that, due to the potentially conflicting nature of the performance metrics (upper/lower task load bound violation and unnecessary loiter), longer processing times do not necessarily translate into inferior

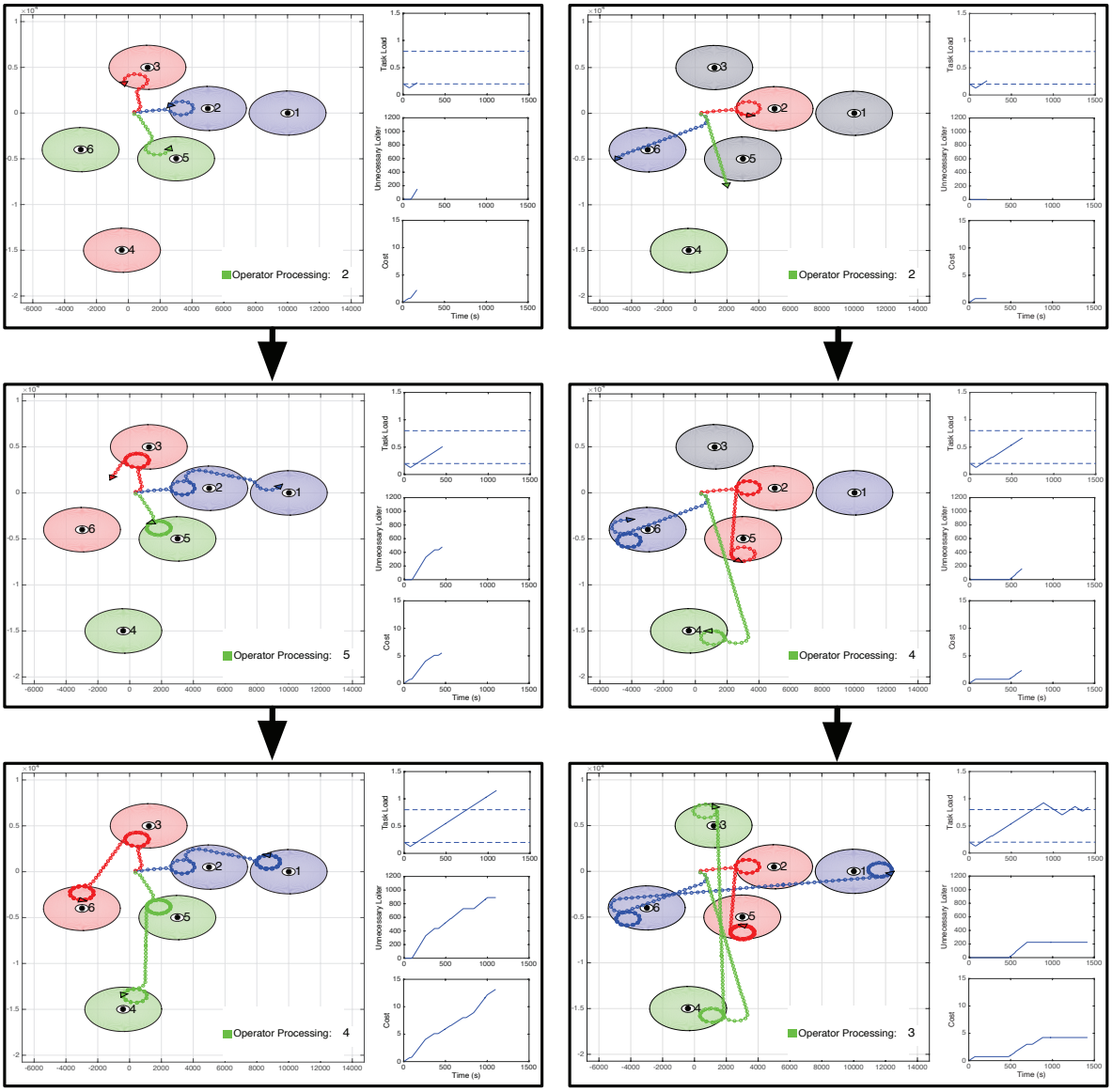

Figure 6.7: An example mission progression for the baseline solution, which ignores task load and resource synchronization issues (left column), and the scenario-based solution (right column).

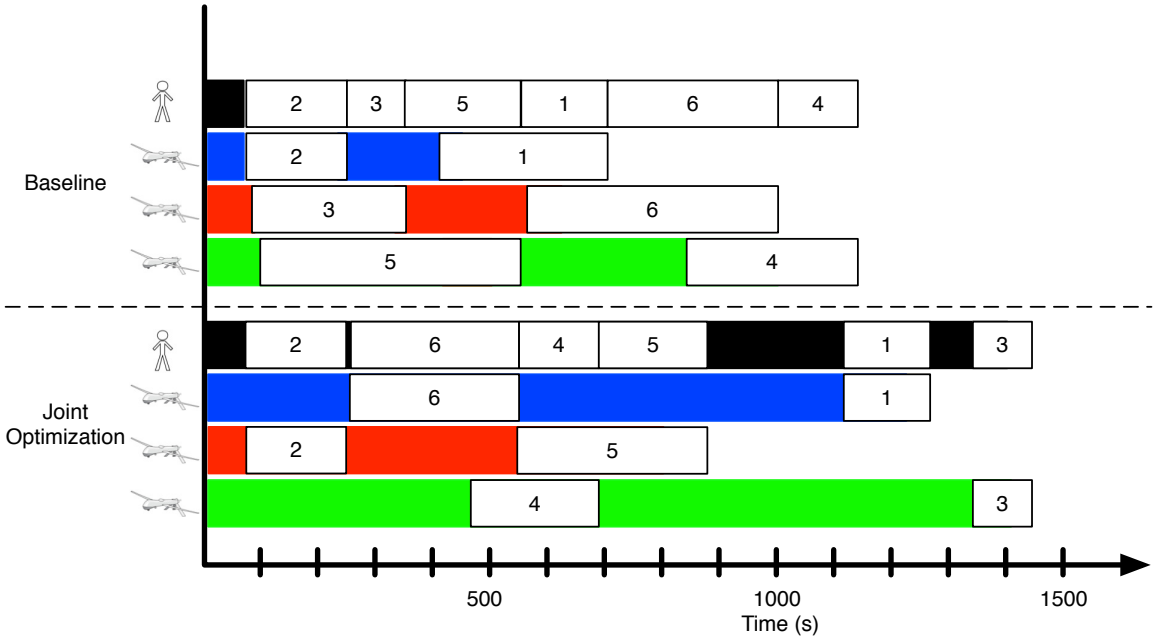

Figure 6.8: Resource utilization during the mission depicted in Figure 6.9. Notice how the scenario-based optimization synchronizes resources to avoid bottlenecks during the mission.

Method	Median	IQR	Whisker Span
Baseline	11.2053	3.2488	11.7385
Expect	3.3812	2.7701	7.7412
"Worst" Case	2.7247	2.1602	6.5284
$Q = 1$	2.9615	2.2192	6.7322
$Q = 5$	2.7821	2.2750	5.6693
$Q = 10$	2.9158	1.9449	5.4648

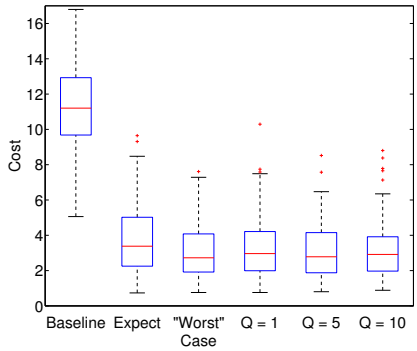

Figure 6.9: Cost statistics obtained over 100 simulation runs for the (i) baseline method, (ii) the dynamic method using "worst"-case processing times, (iii) the dynamic method using expected processing times, and (iv) the dynamic, scenario-based scheme using $Q = 1$, 5, and 10 scenarios.

performance. As such, the 10-scenario method often out-performed the "worst"-case method. As the number of scenarios is increased, one would expect the appearance of poor solutions (outliers) to become less frequent, which generally results in a further decrease in cost variance. This property is an inherent benefit of the scenario-based method: it protects against poor quality solutions even in complex missions where it is not straightforward to assess what conditions will produce the worst-case mission cost. The scenario-based method also allows the user to tune the degree of robustness, by using greater or fewer numbers of scenarios. Note, however, that the increased robustness introduced by higher numbers of scenarios may be at the expense of inferior *mean* performance.

Although the trends illustrated in this example are somewhat typical, it is important to note that the benefit that is gained from implementing a scenario-based approach as oppose to, e.g., choosing expected values, is sensitive to the particular problem at hand. That is, there may be missions in which choosing expected or "worst"-case processing times may actually result in very high-quality solutions, in which case the benefit gained by the scenario-based method may be offset by the added computation. We leave a thorough study of the sensitivity of the scenario-based optimization scheme to problem parameters as a topic of future work.

6.6 Chapter Summary

In many supervisory missions, a tighter coupling between the human and mobile sensor behavior can be obtained by simultaneously coordinating both components within a single planning framework. This chapter explored this concept by developing a joint scheduling and routing framework for a discrete surveillance mission involving a set of static targets, a team of UAVs, and a human operator. The complete operation was formulated as a MINLP, which can be equivalently represented as a MILP of much larger

size. However, when $N = 1$, an equivalent MILP of identical size exists. For scalability, a dynamic heuristic has been developed for constructing solutions within a receding-horizon framework. This framework can provide robustness to uncertainty in processing times by introducing a straightforward scenario-based re-planning operation. Numerical examples illustrated the potential advantages of this joint approach over alternative methods that do not explicitly consider the synchronization of human and autonomous resources.

Future work should include a thorough study of the scenario-based scheme in order to generate performance bounds and quantify its sensitivity to problem parameters. The development of alternative heuristics for solving the full MINLP could also be useful. Adaptations to the optimization framework to allow higher-fidelity task load evolution models could improve performance as well. Finally, validation of the proposed framework on human-test subjects is crucial to further development.

Chapter 7

Conclusions and Future Work

Autonomous mobile sensor teams are already prevalent within a number of current civilian and military applications, and the use of such teams will only grow as technology improves. Despite the maturity of modern hardware and algorithms, however, it is still somewhat rare in practice that these types of autonomous teams operate as completely isolated systems; that is, unmanned or autonomous systems are usually only a small part of a much larger, complex system which usually operates within a dynamic environment and contains diverse system components. In particular, practical engineering systems that utilize autonomous mobile sensors often require some degree of feedback from a human operator in order to function effectively. In this case, the interactions between the human operator and the autonomous agents must be carefully coordinated to avoid bottlenecks and ensure that mission goals are met in an efficient and reliable way.

As a result of the inherent complexities of realistic applications, it is clear that the design process involved with human-centered systems requires the solution of a number of subproblems, each focusing on improving one or multiple system sub-components. The particular design approach usually depends on mission goals, as well as the particular role that the humans and autonomous agents play within the overall system. In this dis-

sertation, we have focused on the development of coordination strategies for *supervisory systems* involving mobile sensors, particular UAVs, in which human operators are responsible for analyzing sensory data generated by the vehicles' on-board cameras. Through a number of illustrative sub-problems, we have demonstrated different approaches to improving overall system effectiveness. Some of these strategies, such as those in Part I, focused primarily on understanding and improving operator behavior by analyzing sensory data, improving user interfaces, and managing data presentation. Other strategies, such as those in Part II, focus on improving the effectiveness of the mobile sensors, by achieving a more balanced workload, preventing undesirable configurations, and optimizing vehicle routes. Finally, in Part III, we studied joint approaches, which sought to simultaneously optimize the vehicle and operator behavior.

We present a brief recap of the main ideas from each chapter here, and finish by discussing some directions of future work.

7.1 Summary

Part I by explored operator-focused methods for improving system performance. We started in Chapter 2 by studying the use of physiological sensing as a means of better understanding operator behavior and interactions with supervisory interfaces. In particular, we presented a very brief pilot usability study in which eye-tracking data was recorded as users interacted with a particular interface for supervisory control of unmanned vehicles (RESCHU). From the data sets, we illustrated how both qualitative and quantitative differences emerged between each user's physiological characteristics during visual search and non-search tasks. Despite the fact that our data did not allow us to draw statistically significant conclusions, this study provided valuable insight into user behavior, demonstrated the processing and analysis of physiological data, and highlighted potential areas

of future research.

In Chapter 3, we studied a different approach to improving human performance in a sequential task analysis mission, which focused on optimizing the operator's schedule, i.e., the order in which the operator processed tasks. We presented a MILP solution approach, which sought to maximize the reward achieved by the operator (or operators) while simultaneously attempting to maintain the task-load imposed on the operator within pre-specified bounds. Using a scenario-based strategy, we showed how robustness to uncertain processing times could be incorporated into the MILP framework. With this in hand, we then illustrated the addition of a number of layers of complexity into the framework to enhance robustness and performance.

Part II shifted our focus to sensor-focused methods for improving the supervisory system. The purpose of the subproblems in this chapter was to illustrate the development of robotic algorithms for common surveillance tasks that arise in supervisory missions. Chapter 4 studied a particular type of persistent surveillance mission in which a team of UAVs is tasked with endlessly surveying a planar region in space for the purpose of detecting some event of interest. We developed a *cloud-based* coverage control strategy to dynamically update agent coverage assignments while only requiring sporadic and unreliable exchanges with a central cloud (repository). This strategy was rigorously analyzed, and shown to have a number of desirable components. In particular, we showed that the coverage regions maintained connectivity, evolved at a timescale appropriate for use within a decomposition-based surveillance framework, and, for certain cases, converged to a configuration that was both Pareto-optimal and a multiplicatively weighted Voronoi partition. Further, when paired with an appropriate trajectory planner, the algorithm provided inherent collision (redundant sensing) avoidance.

In Chapter 5, we studied a different type of surveillance mission, which operates over a discrete region. Here, a single UAV (modeled as a Dubins vehicle) is tasked with pro-

viding visual imagery of a set of targets, each requiring particular imaging behaviors. By placing appropriate restrictions on UAV behaviors, the problem was rigorously posed as a constrained optimization that sought to minimize the total closed tour time, subject to a constraint on the initial maneuver time. We presented a sampling-based approximation of the continuous problem, and developed a heuristic solution method that used solutions to related GTSP instances. We showed how this framework can be used to recover reasonable solutions to the full, continuous routing problem. We then briefly discussed how the single-vehicle solution strategy can be paired with target assignment heuristics in a decomposition-based framework in order to address multi-vehicle problems.

Finally, Part III focused on joint optimization methods for simultaneously constructing UAV routes and operator schedules for a discrete surveillance mission. The subproblem considered in Chapter 6 is essentially a combination of those from Chapters 3 and 5 in a unified framework. Indeed, this chapter considered a mission in which the UAVs were required to visit discrete targets to collect imagery, which was sent to a remotely-located human operator for analysis. We proposed a single, mixed-integer programming-based solution framework for simultaneously constructing vehicle routes and generating the operator's schedule, with the goal of maintaining the operator's task-load within a high-performance regime and reducing unnecessary UAV loitering time. We proved that the full MINLP could be equivalently represented as a MILP, although at the expense of increased complexity in all except the single-vehicle case. We proceeded to develop dynamic heuristics for tractably solving the full problem, and introducing robustness to uncertain operator processing times. We finished by demonstrating the potential advantages of this method in a series of example missions.

7.2 Future Work

The diverse facets of practical, human-centered engineering systems evoke a dense web of design problems that are generally interconnected and span multiple scientific disciplines. As such, multi-disciplinary research is crucial to the maturation of relevant technologies moving forward. In particular, when developing effective human supervisory systems, elements from fields such as control systems, human factors, psychology, operations research, among others, play a crucial role in ensuring that each individual system component is optimized, and can be seamlessly integrated into the overall system.

With regard to supervisory systems, particularly those involving mobile sensors, the broad idea of simultaneously optimizing over operator and autonomous behavior is one that is largely unexplored in the engineering community. Indeed, most existing work in this area assumes only a very "loose" coupling between the two components, e.g. autonomous agent motion is fixed and optimization is performed over operator behavior or vice versa, rather than explicitly addressing the interplay between the two. There is potentially a large benefit to this type of "systems engineering" approach to supervisory design, and thus this reasoning provides a promising avenue of future research.

The work in Part III of this dissertation takes the aforementioned "systems engineering" approach to the design of a discrete persistent surveillance mission; however, there are a number improvements to be made. For example, the framework of Chapter 6 uses a relatively simplistic task-load model, whereas joint optimization frameworks that better capture lower level dynamics of human cognitive processing may boost performance. Physiological sensing, such as eye-tracking, may also be helpful for assessing operator states in real-time, which can, in turn be used within optimization frameworks. However, as suggested by the study in Chapter 2, determining the most beneficial use of eye-tracking within supervisory missions is not straightforward.

In addition, more sophisticated vehicle dynamics could also be integrated into both sensor-focused and joint optimization schemes, and coordination schemes could potentially be developed to better distribute autonomous resources, particularly if the mission contains additional layers of complexity. For example, one could envision an "explore" vs. "exploit" scenario, in which autonomous vehicles are in charge of visiting pre-set targets, but also exploring unknown regions of the map whenever possible. Here, strategic planning is required to determine when and how agents should explore, so as not to create bottlenecks with respect to the overall mission. Future research should also seek to develop similar schemes for autonomous missions outside the realm of surveillance.

There are also many additional considerations that can be added to individual sub-problems in order to enhance realism and to boost performance, both with respect to operator and agent behavior. From a human factors and psychology standpoint, rich research areas include the deeper integration of physiological sensing into supervisory operations and interface design, the incorporation of computer-vision tools and decision aids, and the implementation of adaptive schemes to react to operator behavior. From a control systems standpoint, research efforts should strive to improve coordination strategies for sensor teams, and to develop robust strategies that are equipped to operate within dynamic and unpredictable environments. The summary sections at the end of each chapter provide a number of other suggestions for interesting research avenues with respect to particular sub-problems.

Finally, experimental testing of supervisory control teams will undoubtedly play a crucial role in developing systems for practical use. Indeed, due to the uncertainties that are involved with both human behavior and operational environments, simulations alone often are not enough to verify that a method can be effectively deployed in the field. Therefore, experimental studies will ultimately be a driving factor in understanding the value of human supervisory systems in a given application domain.

Appendix A

Proof of Results from Chapter 4

This chapter provides rigorous proofs of key results from Chapter 4, namely Theorems 1 - 5, Proposition 2, and necessary intermediate results.

We start with one of the aforementioned intermediate results.

Proposition 3 (Sets) *Suppose Assumption 1 holds, and that, at the time of each exchange occurring prior to time $\bar{t} \geq 0$, required algorithmic constructions are well-posed and the cloud performs updates via Algorithm 7. Then, for any $k \in V$ at any time $t \leq \bar{t}$:*

1. $k \in P_{ID_k}$,
2. *k belongs to at most 2 elements of $\boldsymbol{P}$,*
3. *if $\Gamma_{ID_k} = 0$, then $k \notin P_\ell$ for any $\ell \neq ID_k$, and*
4. *if $k \in P_j$, $j \neq ID_k$, then $P_j \cap P_\ell^{ID} = \emptyset$ for $\ell \notin \{j, ID_k\}$*

Proof: Fix $\bar{t} \geq 0$, $k \in V$. When $t = 0$, $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{ID}}$ is an N-partition of V, implying the proposition. Since k is not removed from P_{ID_k} or added to any P_i with $i \neq \mathrm{ID}_k$ until its first *reassignment*, i.e., when ID_k is changed, the proposition is true for all t

prior to the first reassignment. Suppose the proposition holds for all t prior to the p-th reassignment, which occurs at time t_0. Suppose $\mathrm{ID}_k^- = j$, $\mathrm{ID}_k^+ = i \neq j$. Algorithm 7 defines $P_i^{\mathrm{ID},+} = P_i^+ = P_{\mathrm{ID}_k^+}^+$. Thus, $k \in P_{\mathrm{ID}_k^+}^+ = P_i^+$ and remains in these sets until another reassignment. Thus, statement 1 holds for all t prior to the $p+1$-st reassignment. Now note that, by Algorithm 8, reassignment cannot occur at t_0 unless $\Gamma_j^- = 0$. By inductive assumption, statement 3 of the proposition holds when $t = t_0^-$, implying $k \notin P_\ell^-$ for any $\ell \neq j$. Upon reassignment, the timers Γ_j, Γ_i are modified such that $\Gamma_j^+, \Gamma_i^+ > \omega_j^+ + \overline{\Delta} - t_0$. Since (i) ID_k cannot change when $\Gamma_j > 0$, and (ii) agent j exchanges data with the cloud and removes k from P_j prior to time $\omega_j^+ + \overline{\Delta}$, we deduce that k solely belongs to P_j, P_i until the $p+1$-st reassignment. Further, for any $t \geq t_0^+$ at which $\Gamma_i = 0$ and the $p+1$-st reassignment has not yet occurred, $k \in P_i$ exclusively (addition to other sets in $\boldsymbol{P}$ without reassignment is impossible). We deduce statements 2 and 3 for any t prior to the $p+1$-st reassignment. Finally, considering Algorithm 8, it is straightforward to show that $\Gamma_j^- = 0$ implies $P_j^- = P_j^{\mathrm{ID},-}$ ($\Gamma_j = 0$ only if the most recent exchange that manipulated elements of P_j^{ID} involved agent j, after which $P_j = P_j^{\mathrm{ID}}$). Further, (i) no agent claims vertices from P_j^+ unless $\Gamma_j = 0$, and (ii) no vertex is added to a coverage region without reassignment. As such, $P_j \cap P_\ell^{\mathrm{ID}} = \emptyset$ for any $\ell \notin \{j, \mathrm{ID}_k = i\}$ prior to another update in which some other agent claims vertices from P_j. Extending this logic and noting the bound $\underline{\Delta}$, we deduce the same result for any t prior to the $p+1$-st reassignment of k. Noting $\underline{\Delta}$ once again, the proposition follows by induction. ■

Proof of Theorem 1. It suffices to show that Definition 1 is well-posed (Proposition 1) whenever additive sets are required. We proceed by induction. When $t = 0$, $\boldsymbol{P}^{\mathrm{ID}} = \boldsymbol{P}$ is a connected N-partition of V; thus, for any i, $P_i^{\mathrm{ID}} \cap (\bigcup_{j \neq i} P_j) = \emptyset$. The same holds prior to the first agent-cloud exchange, so the first call to Algorithm 7 is well-posed. Now assume that, for all t prior to the p-th call to Algorithm 7, (i) $\boldsymbol{P}^{\mathrm{ID}}$ is a connected N-partition of V,

and (ii) if an exchange that requires P_i^{add} occurs, then $P_i^{\text{ID}} \cap \left(\bigcup_{j\neq i} P_j\right) = \emptyset$. This implies that Proposition 3 holds at any t prior to the p+1-st exchange. Assume the p-th exchange occurs at time t_0 and involves agent i. It is trivial to show $\boldsymbol{P}^{\text{ID},+}$ is an N-partition of V. To show $\boldsymbol{P}^{\text{ID},+}$ is connected, first notice $P_i^{\text{ID},+} = P_i^+$. Since either $P_i^+ = P_i^{\text{add}}(c_i^+)$ (connected by Definition 1) or $P_i^{\text{ID},+} = P_i^{\text{ID},-}$ (connected by assumption), connectivity of $P_i^{\text{ID},+}$ follows. Now consider P_j^{ID}, $j \neq i$. If $\Gamma_j^- \neq 0$, then $P_j^{\text{ID},+} = P_j^{\text{ID},-}$ is connected. Suppose $\Gamma_j^- = 0$ and $P_j^{\text{ID},+}$ is not connected. By Proposition 3, $P_j^+ \cap P_\ell^{\text{ID},+} = \emptyset$ for any $\ell \notin \{i, j\}$. Thus, there exists $k_1 \in P_j^{\text{ID},+}$ such that (i) $k_1 \notin P_i^{\text{add}}(c_i^+)$, and (ii) any optimal path in $G(P_j^+)$ spanning k_1 and c_j^+ contains some $k_2 \in P_i^{\text{add}}(c_i^+) = P_i^+$. Select one such path and vertex k_2. Without loss of generality, assume $\{k_1, k_2\} \in E$. Definition 1 implies $\frac{1}{s_i} d_{P_i^+}(k_2, c_i^+) < \min\{\frac{1}{s_\ell} d_{P_\ell^+}(k_2, c_\ell^+) \,|\, \ell \neq i, k_2 \in P_\ell^+\}$ and thus $\frac{1}{s_i} d_{P_i^+ \cup \{k_1\}}(k_1, c_i^+) < \frac{1}{s_j} d_{P_j^+}(k_1, c_j^+)$. Since $\Gamma_j^- = 0$ and $\text{ID}_{k_1}^- = j$, Proposition 3 implies $\frac{1}{s_i} d_{P_i^+ \cup \{k_1\}}(k_1, c_i^+) < \frac{1}{s_j} d_{P_j^+}(k_1, c_j^+) = \min\{\frac{1}{s_\ell} d_{P_\ell^+}(k_1, c_\ell^+) \,|\, \ell \neq i, k_1 \in P_\ell^+\}$, contradicting $k_1 \notin P_i^{\text{add}}(c_i^+)$. Thus, $P_j^{\text{ID},+}$ is connected. Invoking Proposition 3, the inductive assumption holds for all t prior to the $p + 1^{st}$ exchange, implying well-posedness of the first $p + 1$ exchanges. ■

We proceed to the proof of Theorem 2.

Proof of Theorem 2.

Statement 1: The proof of Theorem 1 implies the statement.

Statement 2: $\boldsymbol{P}$ is an N-covering of V since $\boldsymbol{P}^{\text{ID}}$ is an N-partition of V, and $P_i^{\text{ID}} \subseteq P_i$ for any i (Proposition 3, statement 1). The covering $\boldsymbol{P}$ is connected, since $P_i = P_i^{\text{ID}}$ (connected by statement 1) immediately following any agent-cloud exchange and is unchanged in between updates.

Statement 3: It suffices to show $\text{ID}_{c_i} = i$ for any t, i: this implies $c_i \neq c_j$ for any $i \neq j$, and $c_i \in P_i$ (Proposition 3). Since $\text{ID}_{c_i} = i$ for all i at $t = 0$, the same holds for any t prior to the first agent-cloud exchange. Suppose $\text{ID}_{c_i} = i$ for all i (thus $c_i \neq c_j$ for any

$i \neq j$) prior to the p-th exchange. If agent i is the p-th communicating agent, lines 2, 9 of Algorithm 7 imply $\mathrm{ID}^+_{c_i^+} = i$. Since $d_{P_j^-}(c_j^-, c_j^-) = 0$ for any j, we have $c_j^+ \notin P_i^{\mathrm{add}}(c_i^+)$. Thus, $\mathrm{ID}^+_{c_j^+} = j$, and induction proves the statement.

Statements 4 and 5: Statement 4 follows from (4.1), noting that $P_i^A = P_i$. Statement 5 holds by assumption when $t = 0$. Let $k \in V$, and consider times when ID_k changes (k is *re-assigned*). Since $\mathtt{supp}(\Phi_j^A(\cdot, t)) = P_j = P_j^A$ for any j at $t = 0$, statement 4 implies that, for any t prior to the first reassignment, k belongs exclusively to $\mathtt{supp}(\Phi^A_{\mathrm{ID}_k}(\cdot, t))$. Suppose statement 5 holds for all t prior to the p-th reassignment (occurring at time t_0), and $\mathrm{ID}_k^- = j$, $\mathrm{ID}_k^+ = i \neq j$. Then, $\Gamma_j^- = 0$ and k belongs exclusively to P_j^- when $t = t_0^-$ (Proposition 3). By Algorithm 7 and 8, $k \in P_i^{A,\mathrm{pd},+}$ and $\Gamma_i^+ > \omega_j^+ + \overline{\Delta} - t_0 \geq \gamma_i^{A,+}$. Since $\mathtt{supp}(\Phi_i^A(\cdot, t))$ is unchanging over an interval of length at least $\Gamma_i^+ \geq \gamma_i^{A,+}$, Equation (4.1) implies $k \notin \mathtt{supp}(\Phi_i^{\mathrm{A}}(\cdot, t))$ when $t \in [t_0^+, t_0^+ + \gamma_i^{A,+}]$. Since k is re-assigned when $t = t_0$, $k \in P_i^+ \backslash P_i^{\mathrm{ID},-}$ and $\Gamma_j^+ = \omega_j^+ + \overline{\Delta} - t_0$. Agent j will communicate with the cloud at some time $t_1 < t_0 + \Gamma_j^+ = \omega_j^+ + \overline{\Delta} < t_0 + \Gamma_i^+$. Thus, $\Gamma_i > 0$ when $t = t_1$, and k is removed from both P_j and $\mathtt{supp}(\Phi_j^A(\cdot, t))$. Thus, for all $t > t_0 + \gamma_i^+$ and before the $p+1$-st reassignment, k belongs exclusively to $\mathtt{supp}(\phi_i(\cdot, t))$. ■

Next, we prove Theorem 3.

Proof of Theorem 3. Theorem 2 implies statement 1.

Statement 2: For any i, (i) $\Gamma_i = 0$ when $t = 0$, and (ii) $\frac{1}{s_i} d_{V'}(k_1, k_2) \leq \overline{d}$ for any connected $V' \subseteq V$, $k_1, k_2 \in V'$. Thus, it is straightforward to show, for any i, t, we have the bound $\Gamma_i \leq \overline{\Delta} + \Delta_{\mathrm{H}} + \overline{d}$. We show by induction that, for any i, t, the bound $\gamma_i^A - t + \omega_i^A \leq \Gamma_i - \Delta_{\mathrm{H}}$ also holds: $\Gamma_i = 0$ and $\gamma_i^A = -\Delta_{\mathrm{H}}$ when t = 0 , so $\gamma_i^A - t + \omega_i^A = \gamma_i^A \leq \Gamma_i - \Delta_{\mathrm{H}}$, and the bound holds prior to the first cloud-agent exchange involving *any* agent, since $\gamma_i^A - t = \gamma_i^A - t + \omega_i^A \leq -\Delta_{\mathrm{H}} \leq \Gamma_i - \Delta_{\mathrm{H}}$ at any such time. Assume the bound holds prior to the p-th exchange (occuring at $t = t_0$). Consider 2 cases: if

agent i is the communicating agent, then $\gamma_i^{A,+} - t + \omega_i^A = \gamma_i^+ \coloneqq T_i^+ - \Delta_{\mathrm{H}}$; if not, then $\gamma_i^{A,+} = \gamma_i^{A,-}$ and either 1) $\Gamma_i^- = \Gamma_i^+$ implying the desired bound, or 2) $\Gamma_i^- = 0$ and $\gamma_i^{A,+} - t_0 + \omega_i^{A,+} = \gamma_i^{A,-} - t + \omega_i^{A,-} \leq \Gamma_i^- - \Delta_{\mathrm{H}} = -\Delta_{\mathrm{H}} \leq (\omega_i^{A,+} + \overline{\Delta} - t_0) - \Delta_{\mathrm{H}} = \Gamma_i^+ - \Delta_{\mathrm{H}}$. This logic extends to all t prior to the $p+1$-st exchange and the desired bound follows from an inductive argument.

Using the aforementioned two bounds, we have $\gamma_i^A + \omega_i^A \leq t + \overline{\Delta} + \overline{d}$. Fix t and $k \in \texttt{Proh}_i(t) \cap P_i$. Then, $k \in P_i^{A,+} = P_i^+$, $k \in P_i^{A,\mathrm{pd},+}$, and $t - \omega_i^{A,+} < \gamma_i^{A,+}$ ('+' is with respect to the fixed time t). Further, over the interval $[t, \omega_i^{A,+} + \gamma_i^{A,+}]$, the vertex k is not re-assigned, P_i is not augmented, and γ_i^A is unchanged. Therefore, $k \notin \texttt{Proh}_i(\omega_i^{A,+} + \gamma_i^{A,+})$. Setting $t_0 \coloneqq \omega_i^{A,+} + \gamma_i^{A,+}$, we have $t < t_0 \leq t + \overline{\Delta} + \overline{d}$. Since $\Gamma_i \geq \gamma_i^A + \Delta_{\mathrm{H}}$ at time $\omega_i^{A,+}$, k is not re-assigned during the interval $[\omega_i^{A,+}, \omega_i^{A,+} + T_i^+] \supseteq [\omega_i^{A,+}, t_0 + \Delta_{\mathrm{H}}] \supseteq [t_0, t_0 + \Delta_{\mathrm{H}}]$. Thus $k \in P_i \backslash \texttt{Proh}_i(\cdot)$ over the same interval.

Statement 3: Fix t and suppose $k \in P_i^- \backslash P_i^+$ (in this proof, '$+, -$' are with respect to t). Then, (i) ID_k changed (k was reassigned) at time $t_0 < t$, (ii) agent i exchanges data with the cloud at time t, and (iii) no exchanges involving agent i occurred during the interval $[t_0, t)$. Upon reassignment at time t_0, Algorithm 8 specifies that (i) Γ_i is reset to value $\omega_i^{A,-} + \overline{\Delta} - t_0$, thus P_i^{ID} is unchanged over the interval $[t_0, t)$, (ii) k is added to $P_{\mathrm{ID}_k}^{A,\mathrm{pd}}$, and (iii) $\gamma_{\mathrm{ID}_k}^A$, T_{ID_k} are given values of at least

$$\tilde{\omega} \coloneqq \max_{\tilde{k} \in P_i^- \backslash P_i^+} \left\{ \omega_i^{A,-} + \overline{\Delta} + \frac{1}{s_i} d_{P_i^-} \left(\tilde{k}, P_i^{\mathrm{ID},-} \right) - t_0 \right\},$$

implying that P_{ID_k}, $\texttt{Proh}_{\mathrm{ID}_k}(\cdot)$ remain unchanged over the interval $(t_0, \tilde{\omega}] \supseteq (t_0, t + \frac{1}{s_i} d_{P_i^-}(k, P_i^{\mathrm{ID},-})] \supseteq (t_0, t]$.

Since coverage regions are connected and non-empty (Theorem 2), and $P_i^- \cap P_\ell^{\mathrm{ID}} = \emptyset$ for any $\ell \notin \{i, \mathrm{ID}_k^+\}$ on the interval $(t_0, t]$ (Proposition 3), (i) there exists a path of length $d_{P_i^-}(k, P_i^{\mathrm{ID},-})$ from k into $P_i^{\mathrm{ID},-}$ and every vertex along any such path (except the

terminal vertex) lies within $P_i^-\backslash P_i^+$, and (ii) $P_i^-\backslash P_i^+ \subseteq \texttt{Proh}_{\text{ID}_k^+}$ over the interval $(t_0, \tilde{\omega}] \supseteq [t, t+\frac{1}{s_i}d_{P_i^-}(k, P_i^{\text{ID},-})]$. Since (i) each vertex belongs to no more than two coverage regions (Proposition 3), (ii) $k \in P_i^-\backslash P_i^+$, and (iii) no agent claims vertices in $\texttt{Proh}_{\text{ID}_k^+}(\cdot) \cap P_{\text{ID}_k^+}$ when $\Gamma_{\text{ID}_k^+} > 0$, vertices along the path (excluding the terminal vertex) do not belong to P_j with $j \neq \text{ID}_k$ over $[t, t+\frac{1}{s_i}d_{P_i^-}(k, P_i^{\text{ID},-})]$. To complete the proof, Algorithm 8 implies $\Gamma_i^+ > \frac{1}{s_i}d_{P_i^-}(k, P_i^{\text{ID},-})$, and thus $P_i^{\text{ID},-} \subseteq P_i^{\text{ID}}$ over $[t, t+\frac{1}{s_i}d_{P_i^-}(k, P_i^{\text{ID},-})]$. ■

The following proposition characterizes a key property of the cost $\mathcal{H}$.

Proposition 4 (Cost) *Suppose Assumption 1 holds and that, during each agent-cloud exchange, the cloud updates relevant global and local coverage variables via Algorithm 7. If $\Phi(\cdot, t_1) = \Phi(\cdot, t_2)$ for all t_1, t_2, then $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}^{ID}, \cdot) = \mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, \cdot)$.*

Proof: Since Φ is time-invariant, $\mathcal{H}(\cdot, \cdot, t_1) = \mathcal{H}(\cdot, \cdot, t_2)$ for any t_1, t_2. When $t = 0$, $\boldsymbol{P} = \boldsymbol{P}^{\text{ID}}$ and $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}^{\text{ID}}, 0) = \mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, 0)$. The same is true prior to the first agent-cloud exchange. Suppose that, prior to the p-th exchange (occurring at $t = t_0$, involving agent i), we have $\mathcal{H}(\boldsymbol{c}^-, \boldsymbol{P}^{\text{ID},-}, t_0) = \mathcal{H}(\boldsymbol{c}^-, \boldsymbol{P}^-, t_0)$. Recall that, for any j, P_j and P_j^{ID} coincide immediately following any exchange involving agent j and, if agent j claims vertices from P_i, then Algorithm 8 ensures that agent i exchanges data with the cloud before additional vertices are claimed by other agents. Considering the p-th update, this logic, along with Proposition 3, implies that $P_i^{\text{ID},-} \cap P_j^- = \emptyset$, for all $j \neq i$. Noting that $c_i^+ \in P_i^{\text{ID},-}$, we deduce that any $k \in P_i^{\text{ID},-}$ contributes equivalently to $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^{\text{ID},+}, t_0)$ and $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^+, t_0)$. If $k \in P_i^{\text{add}}(c_i^+)\backslash P_i^{\text{ID},-}$, then for any $j \neq i$ such that $k \in P_j^+$, we have $\frac{1}{s_i}d_{P_i^+}(k, c_i^+) < \frac{1}{s_j}d_{P_j^+}(k, c_j^+)$ (Definition 1), implying k contributes equivalently to $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^{\text{ID},+}, t_0)$ and $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^+, t_0)$. Now suppose $k \in P_j^+\backslash P_i^+$, where $P_j^+ \cap P_i^+ \neq \emptyset$. We show that $d_{P_j^{\text{ID},+}}(c_j^+, k) = d_{P_j^+}(c_j^+, k)$: if a length-minimizing path in $G(P_j^+)$ between c_j^+ and k is also contained in $G(P_j^{\text{ID},+})$, the result is trivial. Suppose that every such minimum length path leaves $G(P_j^{\text{ID},+})$. By Proposition 3, every $\overline{k} \in P_j^+$ must sat-

isfy $\mathrm{ID}_k^+ \in \{i, j\}$. Thus, assume without loss of generality that k is adjacent to P_i^+. Let $\overline{k} \in P_i^+$ be a vertex that is adjacent to k and lies along a minimum-length path in $G(P_j^+)$ spanning c_j^+ and k. Since $\overline{k} \in P_i^+ \backslash P_i^{\mathrm{ID},-}$, we have $\overline{k} \in P_i^{\mathrm{add}}(c_i^+)$ as constructed during the update, implying $\frac{1}{s_i} d_{P_i^+}(\overline{k}, c_i^+) < \min\{\frac{1}{s_\ell} d_{P_\ell^+}(\overline{k}, c_\ell^+) \,|\, \ell \neq i, \overline{k} \in P_\ell^+\}$ and thus $\frac{1}{s_i} d_{P_i^+ \cup \{k\}}(k, c_i^+) < \frac{1}{s_j} d_{P_j^+}(k, c_j^+)$. Since $\Gamma_j^- = 0$ and $\mathrm{ID}_k^- = j$, Proposition 3 implies $\frac{1}{s_i} d_{P_i^+ \cup \{k\}}(k, c_i^+) < \frac{1}{s_j} d_{P_j^+}(k, c_j^+) = \min\{\frac{1}{s_\ell} d_{P_\ell^+}(k, c_\ell^+) \,|\, \ell \neq i, P_\ell^+\}$, contradicting $k \notin P_i^{\mathrm{add}}(c_i^+) \subset P_i^+$. Thus, $d_{P_j^{\mathrm{ID},+}}(c_j^+, k) = d_{P_j^+}(c_j^+, k)$, which, by inductive assumption, implies that k contributes equally to the value of both $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^{\mathrm{ID},+}, t_0)$ and $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^+, t_0)$. We conclude $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^{\mathrm{ID},+}, t_0) = \mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^+, t_0)$. Since $\boldsymbol{P}$, $\boldsymbol{P}^{\mathrm{ID}}$, and $\boldsymbol{c}$ are static between updates, the statement follows by induction. ■

With this result in hand, we are in position to complete the proof of Theorem 4.

Proof of Theorem 4. The cost $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t)$ is static in between agent-cloud exchanges, as $\boldsymbol{P}$ and $\boldsymbol{c}$ are unchanged. Consider an exchange occurring at time t_0 involving agent i. By Proposition 4, we have $\mathcal{H}(\boldsymbol{c}^+, \boldsymbol{P}^+, t_0) \leq \mathcal{H}(\boldsymbol{c}^-, \boldsymbol{P}^{\mathrm{ID},-}, t_0) = \mathcal{H}(\boldsymbol{c}^-, \boldsymbol{P}^-, t_0)$. Thus, the cost $\mathcal{H}(\boldsymbol{c}, \boldsymbol{P}, t)$ is non-increasing in time. Since $\mathrm{Cov}_N(V)$ is finite, there is some time t_0 after which the value of $\mathcal{H}$ is static. Consider fixed $t > t_0$ at which some agent i exchanges data with the cloud. Since the value of $\mathcal{H}$ is unchanging, Algorithm 7 implies that $\boldsymbol{P}^{\mathrm{ID}}$ and $\boldsymbol{c}$ are unchanged by the update. It follows that $\boldsymbol{c}$ and $\boldsymbol{P}^{\mathrm{ID}}$ converge in finite time. Further, since $P_i^{\mathrm{ID}} \subseteq P_i$ for any i (Proposition 3), we have $P_i^{\mathrm{ID},-} = P_i^{\mathrm{ID},+} = P_i^{\mathrm{ID},-} \cup P_i^{\mathrm{add}}(c_i^+) = P_i^+$. By persistence of exchanges imposed by $\overline{\Delta}$, this implies that after some finite time, $\boldsymbol{P}$ and $\boldsymbol{P}^{\mathrm{ID}}$ are concurrent.

To prove Pareto optimality of the limiting configuration, consider t_0, such that for all $t > t_0$, the pair $(\boldsymbol{c}, \boldsymbol{P})$ is unchanging and $\boldsymbol{P}$ is an N-partition of V. Timers are only reset when $\boldsymbol{P}$ is altered, so assume without loss of generality that $\Gamma_i = 0$ for all i at any $t > t_0$. Suppose agent i exchanges data with the cloud at time $t > t_0$. Algorithm 7

implies that there is no $k \in P_i$ such that $\sum_{h\in P_i} d_{P_i}(h,k)\Phi(h,t) < \sum_{h\in P_i} d_{P_i}(h,c_i)\Phi(h,t)$ (if not, the cost is lowered by moving c_i). Similarly, for $k \in P_j$ with $j \neq i$ that is adjacent to P_i, we have $\frac{1}{s_i} d_{P_i\cup\{k\}}(c_i,k) \geq \frac{1}{s_j} d_{P_j}(c_j,k)$ (if not, there exists $k \in P_i^{\text{add}}(c_i^+)\backslash P_i^-$, contradicting convergence). As such, for any i, there is no $V' \subset V\backslash P_i$ such that $\sum_{k\in V'} \frac{1}{s_i} d_{P_i\cup V'}(c_i,k) < \sum_{k\in V'} \min\{\frac{1}{s_j} d_{P_j}(c_j,k) \,|\, k \in P_j, j \neq i\}$, implying statement 2 of Definition 2. ■

We now prove Proposition 2, which relates the notion of Pareto-optimality and that of a (multiplicatively-weighted) centroidal Voronoi partition.

Proof of Proposition 2. Since $\boldsymbol{P}$ is an N-partition of V by assumption, we have

$$\mathcal{H}(\boldsymbol{c},\boldsymbol{P},t) = \sum_{i=1}^{N} \sum_{k\in P_i} d_{P_i}(k,c_i)\Phi(k,t).$$

Thus it is clear that each c_i must be a centroid of the respective region in order for the pair $(\boldsymbol{c},\boldsymbol{P})$ to satisfy property 1 in Definition 2. We show that $\boldsymbol{P}$ is a multiplicatively-weighted Voronoi partition generated by $\boldsymbol{c}$ and weighted by $\boldsymbol{s}$ by contradiction. Suppose that $(\boldsymbol{c},\boldsymbol{P})$ satisfies property 2 of Definition 2 and that $c_i \in C(P_i,t)$ for all i. This implies that for any i, there is no $V' \subset V\backslash P_i$ such that $\sum_{k\in V'} \frac{1}{s_i} d_{P_i\cup V'}(c_i,k) < \sum_{k\in V'} \min\{\frac{1}{s_j} d_{P_j}(c_j,k) | k \in P_j, j \neq i\}$. Now suppose that $\boldsymbol{P}$ is *not* a multiplicatively-weighted Voronoi partition generated by $\boldsymbol{c}$ and weighted by $\boldsymbol{s}$: There exists $k \in P_j$ and $i \neq j$ satisfying $\frac{1}{s_i} d_V(k,c_i) < \frac{1}{s_j} d_V(k,c_j)$. Let $\rho \coloneqq (c_i \coloneqq k_1, k_2, \ldots, k_n \coloneqq k)$ be the shortest path in $G(V)$ spanning k and c_i (notice this also implies $(k_{p_1}, k_{p_1+1}, \ldots, k_{p_2})$ is a length minimizing path spanning k_{p_1} and k_{p_2} for any $p_1, p_2 \in \{1,\ldots,n\}$, $p_1 < p_2$). Let $p \in \{1,\ldots,n\}$ be the first index satisfying $k_p \notin P_i$, i.e., $k_r \in P_i$ for all $r < p$. Assume $k_p \in P_\ell$. Then, $\tilde{\rho} \coloneqq (k_1, k_2, \ldots, k_p)$ is a length minimizing path in $G(V)$ spanning c_i and k_p. Since $k_p \notin P_i$ and is adjacent

to P_i, we have by assumption $\frac{1}{s_\ell} d_{P_\ell}(c_\ell, k_p) \leq \frac{1}{s_i} d_V(c_i, k_p)$. Indeed, if $\frac{1}{s_\ell} d_{P_\ell}(c_\ell, k_p) > \frac{1}{s_i} d_V(c_i, k_p)$, then $\frac{1}{s_\ell} d_{P_\ell}(c_\ell, k_p) > \frac{1}{s_i} d_{P_i \cup \{k_p\}}(c_i, k_p)$ which is a contradiction. If $k_{p+1} \in P_\ell$, then $\frac{1}{s_\ell} d_{P_\ell}(c_\ell, k_{p+1}) \leq \frac{1}{s_i} d_V(c_i, k_{p+1})$. If $k_{p+1} \in P_z$, $z \neq \ell$, then $(c_i \coloneqq k_1, k_2, \ldots, k_{p+1})$ is a length minimizing path between c_i and k_{p+1} and, by similar logic, we must have $\frac{1}{s_z} d_{P_z}(c_z, k_{p+1}) \leq \frac{1}{s_\ell} d_{P_\ell \cup \{k_{p+1}\}}(c_\ell, k_{p+1}) \leq \frac{1}{s_i} d_V(c_i, k_{p+1})$. Building inequalities inductively yields $\frac{1}{s_j} d_V(c_j, k) \leq \frac{1}{s_j} d_{P_j}(c_j, k) \leq \frac{1}{s_i} d_V(c_i, k)$, which is a contradiction. Thus, $\boldsymbol{P}$ must be a multiplicatively-weighted Voronoi partition, generated by $\boldsymbol{c}$ and weighted by $\boldsymbol{s}$. ■

Finally, we conclude the appendix with the proof of Theorem 5.

Proof of Theorem 5. By Assumption 2, local trajectory planners never direct an agent into its prohibited region, so if no exchange occurs that removes the vertex corresponding to the relevant agent's location from its coverage region, the statement is immediate. Suppose, at some t_0, agent i, which is located at $k \in P_i^{A,-}$, exchanges data with the cloud and k is removed, i.e., $k \notin P_i^{A,+}$. At time t_0^+, agent i executes Algorithm 9, lines 5 and 6. Theorem 3 ensures (i) there exists a path in $G(P_i^{A,-})$ between k and $P_i^{\mathrm{ID},-}$, (ii) all vertices along the path belong to $\texttt{Proh}_{\mathrm{ID}_k^+} \setminus \bigcup_{j \neq \mathrm{ID}_k^+} P_j^+$ during the interval $(t_0, t_0 + \frac{1}{s_i} d_{P_i^{A,-}}(k, P_i^{\mathrm{ID},-})]$, and (iii) $P_i^{\mathrm{ID},-} \subseteq P_i \coloneqq P_i^A$ over the same interval. Thus, if agent i immediately moves along the path, it lies exclusively within $\texttt{Proh}_{\mathrm{ID}_k^+}$ until it reaches $P_i^{A,+}$. It remains to show that the agent does not enter $\texttt{Proh}_i(\cdot) \cap P_i^+$ while traversing the aforementioned path. Without loss of generality, consider the update at time t_0 previously described. Since k is re-assigned prior to the update, we have $\texttt{Proh}_i(t_0^-) \cap P_i^- = \emptyset$ (vertices in P_i are not claimed unless $\Gamma_i = 0$, implying $t_0 - \omega_i^{A,-} > \gamma_i^{A,-}$). By Proposition 3, we deduce $\Gamma_{\mathrm{ID}_k^+}^+ > 0$, so no vertices in $P_i^{A,-} \cap P_{\mathrm{ID}_k^+}^+$ can belong to $P_i^{A,\mathrm{pd},+}$, and no vertex on the constructed path belongs to $\texttt{Proh}_i(t_0^+)$. Since $\Gamma_i^+ > \gamma_i^{A,+} > \frac{1}{s_i} d_{P_i^{A,-}}(k, P_i^{\mathrm{ID},-})$, $\texttt{Proh}_i(\cdot)$ remains unchanged over $(t_0, t_0 + \frac{1}{s_i} d_{P_i^{A,-}}(k, P_i^{\mathrm{ID},-})]$. ■

Appendix B

Resolution Completeness of Algorithm 14

In this appendix, we rigorously characterize the *resolution completeness* properties of Algorithm 14. These properties essentially serve to justify the use of the discrete approximation of Problem 2 for the purpose of approximating solutions to Problem 1, by showing that, loosely, the quality of solutions produced by Algorithm 14, in typical cases, improves with finer discretization granularity.

We start with some preliminary notions. For this appendix, we assume that all angles $\theta \in [0, 2\pi)$ are equivalently represented as points on the unit circle $S^1 \coloneqq \{x \in \mathbb{R}^2 \mid \|x\| = 1\}$ via the relation $\theta \mapsto (\cos(\theta), \sin(\theta))$. As such, we assume without loss of generality that the UAV configuration space is $\mathbb{R}^2 \times S^1$. Recall that $\texttt{DWL}_j$ is defined as the set of configurations from which a UAV can begin executing a feasible dwell-time maneuver at T_j. We start by parameterizing the set of feasible UAV initial maneuvers and the set of feasible closed trajectories with respect to Problem 1.

Definition 5 (Initial Maneuver) *The* initial maneuver parameterization set *is de-*

fined

$$\texttt{INL}_\epsilon \coloneqq \left\{ v \in \bigcup_{j=1}^{M} \texttt{DWL}_j \;\middle|\; \texttt{DIST}(v_0, v) \leq \epsilon \right\},$$

where $\texttt{DIST}(v_0, v)$ *is the time required for the UAV to traverse the optimal Dubins path from* v_0 *to* v*.*

Notice that, with slight abuse of notation, we have re-defined $\texttt{INL}_\epsilon$ as the continuous analog of the discrete set of Section 5.2.

Definition 6 (Closed Trajectory) *The* closed trajectory parameterization set *is defined*

$$\texttt{CLS}_\epsilon \coloneqq \left\{ \boldsymbol{v} \in (\texttt{DWL}_{\sigma(1)} \cap \texttt{INL}_\epsilon) \times \texttt{DWL}_{\sigma(2)} \times \cdots \times \texttt{DWL}_{\sigma(M)} \mid \sigma \text{ permutes the set } \{1, \ldots, M\} \right\}.$$

With this definition, we can associate to each $\mathbf{v} \coloneqq (v_1, v_2, \ldots, v_M) \in \texttt{CLS}_\epsilon$ a feasible solution of Problem 1 by (i) appending the initial configuration v_0, (ii) constructing optimal Dubins routes between successive nodes (connecting v_M to v_1), and (iii) appending dwell-time maneuvers. Strictly speaking, the set $\texttt{CLS}_\epsilon$ does not completely parameterize the solution set of Problem 1, since, for some j, there may exist $v \in \texttt{DWL}_j$ that is the starting configuration for multiple distinct dwell-time maneuvers. However, this ambiguity is inconsequential to the present analysis, so we can assume that, for each j, the correspondence between $\texttt{DWL}_j$ and the set of dwell-time maneuvers at T_j is bijective (see Remark 17). We now assign an appropriate cost to each element of $\texttt{CLS}_\epsilon$.

Definition 7 (Cost) *Define the map* $\texttt{LGTH} \colon \texttt{CLS}_\epsilon \to \mathbb{R}_{\geq}$ *where* $\texttt{LGTH}((v_1, v_2, \ldots, v_M))$ *is the time required for the UAV to traverse the closed tour (beginning at* v_1*) that sequentially touches and performs the dwell-time maneuver associated with all vertices* $v_1, \ldots, v_M$*.*

For each T_j, the length of an appropriate dwell-time maneuver varies continuously as a function of the maneuver's starting configuration (element in $\texttt{DWL}_j$). As such, results in [124] imply that (i) the map LGTH is well-defined and continuous except on a finite set of $(3M-1)$-dimensional smooth surfaces embedded in $(\mathbb{R}^2 \times S^1)^M$, and (ii) in the limit from one side of each discontinuity surface, LGTH is continuous up to and on the surface. The presence of discontinuity sets necessitates the following definition. Here, $\texttt{LGTH}^* \coloneqq \min \texttt{LGTH}$.

Definition 8 (Degeneracy) *An instance of Problem 1 is called* degenerate *if, for every sequence $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \in \boldsymbol{CLS}_\epsilon$ such that $\boldsymbol{LGTH}(\boldsymbol{v}_k) \to \boldsymbol{LGTH}^*$ and $\boldsymbol{v}^* \coloneqq \lim_{k\to\infty} \boldsymbol{v}_k$ exists, the limit $\boldsymbol{v}^*$ either (i) is not contained in $\boldsymbol{CLS}_\epsilon$, (ii) belongs to a discontinuity set of $\boldsymbol{LGTH}$, or (iii) has a first component (corresponding to the starting point of the closed trajectory) that is isolated in the set INL_ϵ.*

Finally, we introduce the notion of a *dense sampling procedure*.

Definition 9 (Dense Sampling Procedure) *Suppose that, for each $N_s \in \mathbb{N}$, a set $\mathcal{A} \subset \mathbb{R}^2 \times S^1$ is sampled N_s times to form a discrete subset A_{N_s}, i.e. $|A_{N_s}| = N_s$ for each N_s. Such a procedure is called a* dense sampling procedure *if, for any $a \in \mathcal{A}$ and any open neighborhood U of a, there exists $\hat{N}_s \in \mathbb{N}$ such that $A_{N_s} \cap U$ is non-empty for all $N_s > \hat{N}_s$.*

We are now ready to rigorously characterize resolution completeness of Algorithm 14.

Theorem 9 (Resolution Completeness) *If Problem 1 is not degenerate, and the set $\boldsymbol{INL}_\epsilon$ is such that: (i) $\boldsymbol{INL}_\epsilon \neq \emptyset$, and (ii) there exists $\hat{\jmath} \in \{1, \ldots, M\}$ such that either $\boldsymbol{INL}_\epsilon \subseteq \boldsymbol{DWL}_{\hat{\jmath}}$ or $\boldsymbol{DWL}_{\hat{\jmath}} \subseteq \boldsymbol{INL}_\epsilon$, then Algorithm 14 is* resolution complete *in the following sense:*

Construct a sequence $\{(\boldsymbol{INL\_MNVR}_{N_s}, \boldsymbol{CLS\_TRAJ}_{N_s})\}_{N_s \in \mathbb{N}}$, where each element $(\boldsymbol{INL\_MNVR}_{N_s}, \boldsymbol{CLS\_TRAJ}_{N_s})$ represents the output of a call to Algorithm 14 when: (i) the

number of discrete samples at each target is N_s, (ii) $\texttt{INL}_\epsilon^*$ (Algorithm 13, line 3) is chosen to satisfy the conditions of Theorem 7, and (iii) an optimal GTSP solution is found (Algorithm 13, line 5). Then, $(\texttt{INL\_MNVR}_{N_s}, \texttt{CLS\_TRAJ}_{N_s})$ is a feasible solution to Problem 1 for each N_s, and if a dense sampling procedure is used to generate the discrete node sets at each target, then the length of the tours in the sequence $\{\texttt{CLS\_TRAJ}_{N_s}\}$ approaches the length of an optimal solution to (5.1) *as $N_s \to \infty$.*

Proof: First note that the notion of resolution completeness described here is well-defined under the specified assumptions. Indeed, if the (non-discrete) set $\texttt{INL}_\epsilon$ satisfies the necessary assumptions, then the discrete analog always has the properties required for Theorem 7 to hold, and $\texttt{INL}_\epsilon^*$ can be chosen accordingly when constructing the sequence $\{(\texttt{INL\_MNVR}_{N_s}, \texttt{CLS\_TRAJ}_{N_s})\}_{N_s \in \mathbb{N}}$.

Feasibility of each $\texttt{INL\_MNVR}_{N_s}$ follows from Theorem 6, and the fact that feasible solutions to Problem 2 map to feasible solutions to Problem 1.

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots \in \texttt{CLS}_\epsilon$ be a sequence such that $\texttt{LGTH}(\mathbf{v}_i) \to \texttt{LGTH}^*$. Note that $\texttt{CLS}_\epsilon \subseteq (\mathbb{R}^2 \times S^1)^M$ can be represented as a bounded subset in $(\mathbb{R}^2 \times S^1)^M \subseteq \mathbb{R}^{4M}$. Invoking the Bolzano-Weierstrass theorem and compactness of S^1, any sequence in $\texttt{CLS}_\epsilon$ contains a subsequence that converges to some point in $(\mathbb{R}^2 \times S^1)^M$. Thus, recalling that Problem 1 is non-degenerate, we can assume without loss of generality that $\mathbf{v}^* = \lim_{i \to \infty} \mathbf{v}_i$ exists, is contained in $\texttt{CLS}_\epsilon$, does not belong to a discontinuity sets of $\texttt{LGTH}$, and has a first component that is not isolated in $\texttt{INL}_\epsilon$. Since the function $\texttt{LGTH}$ is continuous at the point $\mathbf{v}^*$, for any $\delta > 0$, there exists an open neighborhood $U \subset \{(\texttt{DWL}_{\sigma(1)} \cap \texttt{INL}_\epsilon) \times \cdots \times \texttt{DWL}_{\sigma(M)} \mid \sigma \text{ permutes the set } \{1, \ldots, M\}\}$ of $\mathbf{v}^*$, within which $\texttt{LGTH}$ takes values within δ of $\texttt{LGTH}^*$. Since the sampling procedure is dense, for some $\hat{N}_s$, there will be a discrete node placed inside of the set U for all $N_s \geq \hat{N}_s$. Since the GTSP is solved exactly, it follows that $|\texttt{LGTH}(\texttt{CLS\_TRAJ}_{N_s}) - \texttt{LGTH}^*| < \delta$ for $N_s > \hat{N}_s$, proving convergence. ■

Theorem 9 states that, under a non-degeneracy assumption about the problem structure, a reasonable implementation of Algorithm 14 will produce solutions that approximate the optimal solutions to the full, multi-objective routing problem. Indeed, the resolution completeness property ensures that Problem 2 will be appropriately reflective of Problem 1.

A few final comments are in order. First, note that degenerate problems occur only in very select situations: a degenerate problem instance is made non-degenerate by perturbing target locations by an arbitrarily small amount (example degenerate problem instances for particular cases are shown in [124]). Thus, the non-degeneracy condition in Theorem 9 is not typically restrictive. Second, if $\texttt{INL}_\epsilon$ does not satisfy the conditions of Theorem 9, then resolution completeness does not hold in general since Algorithm 13 is not guaranteed to find an optimal solution to Problem 2 (Theorem 7). Nevertheless, the quality of solutions produced by Algorithm 14 generally increase as sampling granularity is made increasingly fine. Finally, it is not generally possible to find optimal solutions to GTSPs. Therefore, the utility of Theorem 9 is its ability to provide intuition about qualitative solution behavior, and to ensure that the discrete method herein is an appropriate approximation.

Bibliography

[1] R. Patel, P. Frasca, J. W. Durham, R. Carli, and F. Bullo, *Dynamic partitioning and coverage control with asynchronous one-to-base-station communication*, *IEEE Transactions on Control of Network Systems* **3** (2016), no. 1 24–33.

[2] J. Roberts, *Special issue on uninhabited aerial vehicles*, *Journal of Field Robotics* **23** (2006), no. 3–4.

[3] K. P. Valavanis, *Advances in Unmanned Aerial Vehicles: State Of The Art And the Road To Autonomy*. Springer, 2008.

[4] US Air Force, *Report on technology horizons, a vision for Air Force Science And Technology during 2010–2030*, tech. rep., AF/ST-TR-10-01-PR, United States Air Force., 2010. Retrieved from http://www.defenseinnovationmarketplace.mil/resources/AF_TechnologyHorizons2010-2030.pdf on Feb. 8, 2016.

[5] W. M. Bulkeley, "Chicago's camera network is everywhere." The Wall Street Journal, November 17,, 2009.

[6] C. Drew, "Military taps social networking skills." The New York Times, June 7,, 2010.

[7] T. Shanker and M. Richtel, "In new military, data overload can be deadly." The New York Times, January 16,, 2011.

[8] E. Guizzo, "Obama commanding robot revolution announces major robotics initiative." IEEE Spectrum, June, 2011.

[9] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. MIT press, 1992.

[10] M. Lind, *Plant modelling for human supervisory control*, *Transactions of the Institute of Measurement and Control* **21** (1999), no. 4-5 171–180.

[11] M. A. Goodrich and M. L. Cummings, *Human factors perspective on next generation unmanned aerial systems*, *Handbook of Unmanned Aerial Vehicles* (2015) 2405–2423.

[12] S. R. Dixon and C. D. Wickens, *Automation reliability in unmanned aerial vehicle control: A reliance-compliance model of automation dependence in high workload*, *Human Factors: The Journal of the Human Factors and Ergonomics Society* **48** (2006), no. 3 474–486.

[13] R. Parasuraman, M. Barnes, K. Cosenzo, and S. Mulgund, *Adaptive automation for human-robot teaming in future command and control systems*, tech. rep., Defense Technical Information Center (DTIC), 2007. Available at `http://www.dtic.mil/dtic`.

[14] A. Poole and L. J. Ball, *Eye tracking in hci and usability research*, *Encyclopedia of Human Computer Interaction* **1** (2006) 211–219.

[15] R. B. Towal, M. Mormann, and C. Koch, *Simultaneous modeling of visual saliency and value computation improves predictions of economic choice*, *Proceedings of the National Academy of Sciences* **110** (2013), no. 40 E3858–E3867.

[16] S. Djamasbi, M. Siegel, and T. Tullis, *Generation Y, web design, and eye tracking*, *International Journal of Human-Computer Studies* **68** (2010), no. 5 307–323.

[17] A. T. Duchowski, *A breadth-first survey of eye-tracking applications*, *Behavior Research Methods, Instruments, & Computers* **34** (2002), no. 4 455–470.

[18] C. E. Nehme, *Modeling Human Supervisory Control in Heterogeneous Unmanned Vehicle Systems*. PhD thesis, Department of Aeronautics and Astronautics, MIT, Feb., 2009.

[19] M. L. Cummings and P. Mitchell, *Automated scheduling decision support for supervisory control of multiple UAVs*, *Journal of Aerospace Computing, Information, and Communication* **3** (2006), no. 6 294–308.

[20] M. Freed, R. Harris, and M. Shafto, *Human-interaction challenges in UAV-based autonomous surveillance*, in *Proceedings of the 2004 Spring Symposium on Interactions Between Humans and Autonomous Systems Over Extended Operations*, 2004.

[21] A. Pereira, H. Heidarsson, C. Oberg, D. Caron, B. Jones, and G. Sukhatme, *A communication framework for cost-effective operation of AUVs in coastal regions*, in *Field and Service Robotics* (A. Howard, K. Iagnemma, and A. Kelly, eds.), vol. 62 of *Tracts in Advanced Robotics*, pp. 433–442. Springer, 2010.

[22] R. C. Shah, S. Roy, S. Jain, and W. Brunette, *Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks*, *Ad Hoc Networks* **1** (2003), no. 2-3 215–233.

[23] US Office of the Secretary of Defense, *Unmanned aircraft systems (UAS) roadmap, 2005-2030*, 2005.

[24] L. J. Sorensen, K. I. Overgaard, and T. J. S. Martinsen, *Understanding human decision making during critical incidents in dynamic positioning*, in *Contemporary Ergonomics and Human Factors 2014* (S. Sharples and S. Shorrock, eds.), p. 359, CRC Press, 2014. Proceedings of the International Conference on Ergonomics & Human Factors, Southampton, UK, 7-10 April 2014.

[25] D. C. Klein, *Using Adaptive Automation to Increase Operator Performance and Decrease Stress in a Satellite Operations Environment*. PhD thesis, Colorado Technical University, 2014.

[26] V. A. Banks, N. A. Stanton, and C. Harvey, *Sub-systems on the road to vehicle automation: Hands and feet free but not mind free driving*, *Safety Science* **62** (2014) 505–514.

[27] B. Terwilliger, D. Vincenzi, D. Ison, K. Witcher, D. Thirtyacre, and A. Khalid, *Influencing factors for use of unmanned aerial systems in support of aviation accident and emergency response*, *Journal of Automation and Control Engineering* **3** (2015), no. 3.

[28] U. E. Franke, *Drones, drone strikes, and us policy: The politics of unmanned aerial vehicles*, *Parameters* **44** (2014), no. 1 121.

[29] S. M. Astley, *Evaluation of computer-aided detection (CAD) prompting techniques for mammography*, *The British Journal of Radiology* **78** (2014), no. suppl_1 S20–S25.

[30] W. Barfield and T. A. Dingus, *Human Factors in Intelligent Transportation Systems*. Psychology Press, 2014.

[31] C. Nehme, B. Mekdeci, J. W. Crandall, and M. L. Cummings, *The impact of heterogeneity on operator performance in futuristic unmanned vehicle systems*, *The International C2 Journal* **2** (2008), no. 2 1–30.

[32] L. F. Bertuccelli, N. Pellegrino, and M. L. Cummings, *Choice modeling of relook tasks for UAV search missions*, in *American Control Conference*, (Baltimore, MD, USA), pp. 2410–2415, June, 2010.

[33] L. F. Bertuccelli, N. W. M. Beckers, and M. L. Cummings, *Developing operator models for UAV search scheduling*, in *AIAA Conf. on Guidance, Navigation and Control*, (Toronto, Canada), Aug., 2010.

[34] K. Savla, C. Nehme, T. Temple, and E. Frazzoli, *On efficient cooperative strategies between UAVs and humans in a dynamic environment*, in *AIAA Conf. on Guidance, Navigation and Control*, (Honolulu, HI, USA), 2008.

[35] K. Savla, T. Temple, and E. Frazzoli, *Human-in-the-loop vehicle routing policies for dynamic environments*, in *IEEE Conf. on Decision and Control*, (Cancún, México), pp. 1145–1150, Dec., 2008.

[36] J. W. Crandall, M. L. Cummings, M. Della Penna, and P. M. A. de Jong, *Computing the effects of operator attention allocation in human control of multiple robots*, *IEEE Transactions on Systems, Man & Cybernetics. Part A: Systems & Humans* **41** (2011), no. 3 385–397.

[37] K. Savla and E. Frazzoli, *Maximally stabilizing task release control policy for a dynamical queue*, *IEEE Transactions on Automatic Control* **55** (2010), no. 11 2655–2660.

[38] K. Savla and E. Frazzoli, *A dynamical queue approach to intelligent task management for human operators*, *Proceedings of the IEEE* **100** (2012), no. 3 672–686.

[39] N. D. Powel and K. A. Morgansen, *Multiserver queueing for supervisory control of autonomous vehicles*, in *American Control Conference*, (Montréal, Canada), pp. 3179–3185, June, 2012.

[40] V. Srivastava, R. Carli, C. Langbort, and F. Bullo, *Attention allocation for decision making queues*, *Automatica* **50** (2014), no. 2 378–388.

[41] V. Srivastava and F. Bullo, *Knapsack problems with sigmoid utility: Approximation algorithms via hybrid optimization*, *European Journal of Operational Research* **236** (2014), no. 2 488–498.

[42] M. Majji and R. Rai, *Autonomous task assignment of multiple operators for human robot interaction*, in *American Control Conference*, (Washington, DC), pp. 6454–6459, June, 2013.

[43] V. Srivastava, A. Surana, and F. Bullo, *Adaptive attention allocation in human-robot systems*, in *American Control Conference*, (Montréal, Canada), pp. 2767–2774, June, 2012.

[44] V. Srivastava, F. Pasqualetti, and F. Bullo, *Stochastic surveillance strategies for spatial quickest detection*, *International Journal of Robotics Research* **32** (2013), no. 12 1438–1458.

[45] V. Srivastava, K. Plarre, and F. Bullo, *Randomized sensor selection in sequential hypothesis testing*, *IEEE Transactions on Signal Processing* **59** (2011), no. 5 2342–2354.

[46] K. Cosenzo, R. Parasuraman, and E. De Visser, *Automation strategies for facilitating human interaction with military unmanned vehicles*, in *Human-robot Interactions in Future Military Operations* (M. Barnes and F. Jentsch, eds.), pp. 103–124. Ashgate Publishing, 2010.

[47] M. Scerbo, *Adaptive automation*, in *Neuroergonomics: The Brain At Work* (R. Parasuraman and M. Rizzo, eds.), pp. 239–252. Oxford University Press, 2001.

[48] K. M. Feigh, M. C. Dorneich, and C. C. Hayes, *Toward a characterization of adaptive systems a framework for researchers and system designers*, *Human Factors: The Journal of the Human Factors and Ergonomics Society* **54** (2012), no. 6 1008–1024.

[49] P. R. Murphy, J. Vandekerckhove, and S. Nieuwenhuis, *Pupil-linked arousal determines variability in perceptual decision making*, *PLOS Computational Biology* **10** (2014), no. 9 e1003854.

[50] T. Judd, K. Ehinger, F. Durand, and A. Torralba, *Learning to predict where humans look*, in *Computer Vision, 2009 IEEE 12th international conference on*, (Kyoto, Japan), pp. 2106–2113, IEEE, 2009.

[51] A. Borji and L. Itti, *Defending yarbus: Eye movements reveal observers' task*, *Journal of Vision* **14** (2014), no. 3 29.

[52] R. Peters and L. Itti, *Beyond bottom-up: Incorporating task-dependent influences into a computational model of spatial attention*, in *Computer Vision and Pattern Recognition, 2007. IEEE Conference on*, (Minneapolis, MN, USA), pp. 1–8, IEEE, 2007.

[53] G. Zelinsky, W. Zhang, and D. Samaras, *Eye can read your mind: Decoding eye movements to reveal the targets of categorical search tasks*, *Journal of Vision* **8** (2008), no. 6 380–380.

[54] M. A. Recarte and L. M. Nunes, *Mental workload while driving: effects on visual search, discrimination, and decision making.*, *Journal of Experimental Psychology: Applied* **9** (2003), no. 2 119.

[55] W. Einhäuser, J. Stout, C. Koch, and O. Carter, *Pupil dilation reflects perceptual selection and predicts subsequent stability in perceptual rivalry*, *Proceedings of the National Academy of Sciences* **105** (2008), no. 5 1704–1709.

[56] T. de Greef, H. Lafeber, H. van Oostendorp, and J. Lindenberg, *Eye movement as indicators of mental workload to trigger adaptive automation*, in *Foundations of augmented cognition. neuroergonomics and operational neuroscience*, pp. 219–228. Springer, 2009.

[57] B. P. Bailey and S. T. Iqbal, *Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management*, *ACM Transactions on Computer-Human Interaction (TOCHI)* **14** (2008), no. 4 21.

[58] L. Fern and R. J. Shively, *A comparison of varying levels of automation on the supervisory control of multiple UASs*, in *AUVSI's Unmanned Systems North America*, (Washington, DC, USA), Aug., 2009.

[59] V. Srivastava, A. Surana, M. P. Eckstein, and F. Bullo, *Mixed human-robot team surveillance*, *arXiv preprint arXiv:1311.2796* (2013).

[60] M. L. Cummings, L. F. Bertucelli, J. Macbeth, and A. Surana, *Task versus vehicle-based control paradigms in multiple unmanned vehicle supervision by a single operator*, *IEEE Transactions on Human-Machine Systems* **44** (2014), no. 3 353–361.

[61] D. Gartenberg, L. A. Breslow, J. Park, J. M. McCurry, and J. G. Trafton, *Adaptive automation and cue invocation: The effect of cue timing on operator error*, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (Paris, France), pp. 3121–3130, ACM, 2013.

[62] M. Pickett, D. W. Aha, and J. G. Trafton, *Acquiring user models to test automated assistants.*, in *FLAIRS Conference*, (Orlando, FL , USA), pp. 112–117, AAAI, 2013.

[63] C. Heitmeyer, M. Pickett, L. Breslow, D. Aha, J. G. Trafton, and E. Leonard, *High assurance human-centric decision systems*, in *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2013 2nd International Workshop on*, (San Francisco, CA, USA), pp. 35–41, IEEE, 2013.

[64] E. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, *Handbooks in operations research and management science* **4** (1993) 445–522.

[65] T. Yamada and R. Nakano, *Job shop scheduling*, *IEE control Engineering series* (1997) 134–134.

[66] B. Alidaee and H. Li, *Parallel machine selection and job scheduling to minimize sum of machine holding cost, total machine time costs, and total tardiness costs*, *Automation Science and Engineering, IEEE Transactions on* **11** (2014), no. 1 294–301.

[67] S. Mason and K. Oey, *Scheduling complex job shops using disjunctive graphs: A cycle elimination procedure*, *International Journal of Production Research* **41** (2003), no. 5 981–994.

[68] R. Cheng, M. Gen, and Y. Tsujimura, *A tutorial survey of job-shop scheduling problems using genetic algorithms-I. Representation*, *Computers & Industrial Engineering* **30** (1996), no. 4 983–997.

[69] D. Ouelhadj and S. Petrovic, *A survey of dynamic scheduling in manufacturing systems*, *Journal of Scheduling* **12** (2009), no. 4 417–431.

[70] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton University Press, 2009.

[71] M. Zimmerman, *Task load*, in *Encyclopedia of Clinical Neuropsychology* (J. Kreutzer, J. DeLuca, and B. Kaplan, eds.), pp. 2469–2470. Springer, 2011.

[72] M. L. Cummings and P. J. Mitchell, *Operator scheduling strategies in supervisory control of multiple UAVs*, *Aerospace Science and Technology* **11** (2007), no. 4 339–348.

[73] D. Broadbent, *Differences and interactions between stresses*, *Quarterly Journal of Experimental Psychology* **15** (1963), no. 3 205–211.

[74] P. A. Hancock, *A dynamic model of stress and sustained attention*, *Human Factors: The Journal of the Human Factors and Ergonomics Society* **31** (1989), no. 5 519–537.

[75] M. Westman and D. Eden, *The inverted-U relationship between stress and performance: A field study*, *Work & Stress* **10** (1996), no. 2 165–173.

[76] K. H. Teigen, *Yerkes-Dodson: A law for all seasons*, *Theory & Psychology* **4** (1994), no. 4 525–547.

[77] S. Rathinam, R. Sengupta, and S. Darbha, *A resource allocation algorithm for multi-vehicle systems with non holonomic constraints*, *IEEE Transactions on Automation Sciences and Engineering* **4** (2007), no. 1 98–104.

[78] D. B. Kingston, R. W. Beard, and R. S. Holt, *Decentralized perimeter surveillance using a team of UAVs*, *IEEE Transactions on Robotics* **24** (2008), no. 6 1394–1404.

[79] J. Las Fargeas, P. Kabamba, and A. Girard, *Cooperative surveillance and pursuit using unmanned aerial vehicles and unattended ground sensors*, *Sensors* **15** (2015), no. 1 1365–1388.

[80] C. A. Rabbath, C. Y. Su, and A. Tsourdos, *Guest editorial introduction to the special issue on multivehicle systems cooperative control with application*, *IEEE Transactions on Control Systems Technology* **15** (2007), no. 4 599–600.

[81] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*. Springer, 2007.

[82] K. Savla, E. Frazzoli, and F. Bullo, *Traveling Salesperson Problems for the Dubins vehicle*, *IEEE Transactions on Automatic Control* **53** (2008), no. 6 1378–1391.

[83] J. J. Enright, K. Savla, E. Frazzoli, and F. Bullo, *Stochastic and dynamic routing problems for multiple UAVs*, *AIAA Journal of Guidance, Control, and Dynamics* **34** (2009), no. 4 1152–1166.

[84] W. Malik, S. Rathinam, and S. Darbha, *An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem*, *Operations Research Letters* **35** (2007), no. 6 747–753.

[85] M. Niendorf, P. T. Kabamba, and A. R. Girard, *Stability of solutions to classes of traveling salesman problems*, *IEEE Transactions on Cybernetics* **46** (2016), no. 4.

[86] M. Alighanbari and J. P. How, *A robust approach to the UAV task assignment problem*, *International Journal on Robust and Nonlinear Control* **18** (2008), no. 2 118–134.

[87] K. Helsgaun, *An effective implementation of the Lin–Kernighan traveling salesman heuristic*, *European Journal of Operational Research* **126** (2000), no. 1 106–130.

[88] R. Patel, P. Agharkar, and F. Bullo, *Robotic surveillance and Markov chains with minimal weighted Kemeny constant*, *IEEE Transactions on Automatic Control* **60** (2015), no. 12 3156–3157.

[89] M. Younis and K. Akkaya, *Strategies and techniques for node placement in wireless sensor networks: A survey*, *Ad Hoc Networks* **6** (2008), no. 4 621–655.

[90] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, *Dynamic vehicle routing for robotic systems*, *Proceedings of the IEEE* **99** (2011), no. 9 1482–1504.

[91] N. Mathew, S. L. Smith, and S. L. Waslander, *Multirobot rendezvous planning for recharging in persistent tasks*, *IEEE Transactions on Robotics* **31** (2015), no. 1 128–142.

[92] F. Bullo, R. Carli, and P. Frasca, *Gossip coverage control for robotic networks: Dynamical systems on the space of partitions*, *SIAM Journal on Control and Optimization* **50** (2012), no. 1 419–447.

[93] J. G. Carlsson, *Dividing a territory among several vehicles*, *INFORMS Journal on Computing* **24** (2012), no. 4 565–577.

[94] J. Peng and S. Akella, *Coordinating multiple robots with kinodynamic constraints along specified paths*, *International Journal of Robotics Research* **24** (2005), no. 4 295–310.

[95] B. P. Gerkey and M. J. Matarić, *Sold!: Auction methods for multirobot coordination*, *IEEE Transactions on Robotics and Automation* **18** (2002), no. 5 758–768.

[96] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. No. 1 in Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[97] J. H. Reif and H. Wang, *Social potential fields: A distributed behavioral control for autonomous robots*, *Robotics & Autonomous Systems* **27** (1999), no. 3 171–194.

[98] N. Kariotoglou, D. Raimondo, S. J. Summers, and J. Lygeros, *Multi-agent autonomous surveillance: A framework based on stochastic reachability and hierarchical task allocation*, *ASME Journal of Dynamic Systems, Measurement, and Control* **137** (2015), no. 3 031008.

[99] T. Shima and P. R. Pagilla, *Special issue on analysis and control of multi-agent dynamic systems*, *ASME Journal of Dynamic Systems, Measurement, and Control* **129** (2007), no. 5 569–570.

[100] N. Nigam, *The multiple unmanned air vehicle persistent surveillance problem: A review*, *Machines* **2** (2014), no. 1 13–72.

[101] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre, *Recent advances on multi-agent patrolling*, in *Advances in Artificial Intelligence*, vol. 3171 of *Lecture Notes in Computer Science*, pp. 474–483. Springer, 2004.

[102] F. Pasqualetti, F. Zanella, J. R. Peters, M. Spindler, R. Carli, and F. Bullo, *Camera network coordination for intruder detection*, *IEEE Transactions on Control Systems Technology* **22** (2014), no. 5 1669–1683.

[103] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2 ed., 2000.

[104] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, *Coverage control for mobile sensing networks*, *IEEE Transactions on Robotics and Automation* **20** (2004), no. 2 243–255.

[105] R. Patel, P. Frasca, and F. Bullo, *Centroidal area-constrained partitioning for robotic networks*, *ASME Journal of Dynamic Systems, Measurement, and Control* **136** (2014), no. 3 031024–031024–8.

[106] J. Cortés, *Coverage optimization and spatial load balancing by robotic sensor networks*, *IEEE Transactions on Automatic Control* **55** (2010), no. 3 749–754.

[107] P. O. Fjällström, *Algorithms for graph partitioning: A survey*, *Linköping Electronic Articles in Computer and Information Science* **3** (1998), no. 10.

[108] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, *Discrete partitioning and coverage control for gossiping robots*, *IEEE Transactions on Robotics* **28** (2012), no. 2 364–378.

[109] G. Laporte, *The vehicle routing problem: An overview of exact and approximate algorithms*, *European Journal of Operational Research* **59** (1992), no. 3 345–358.

[110] P. Toth and D. Vigo, eds., *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, 2001.

[111] J. Ousingsawat and M. G. Earl, *Modified lawn-mower search pattern for areas comprised of weighted regions*, in *American Control Conference*, (New York, USA), pp. 918–923, July, 2007.

[112] D. E. Soltero, M. Schwager, and D. Rus, *Generating informative paths for persistent sensing in unknown environments*, in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (Vilamoura, Portugal), pp. 2172–2179, Oct., 2012.

[113] X. Lan and M. Schwager, *Planning periodic persistent monitoring trajectories for sensing robots in Gaussian random fields*, in *IEEE Int. Conf. on Robotics and Automation*, (Karlsruhe, Germany), pp. 2415–2420, May, 2013.

[114] G. Mathew and I. Mezić, *Metrics for ergodicity and design of ergodic dynamics for multi-agent systems*, *Physica D: Nonlinear Phenomena* **240** (2011), no. 4 432–442.

[115] J. F. Araujo, P. B. Sujit, and J. B. Sousa, *Multiple UAV area decomposition and coverage*, in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, (Singapore), pp. 30–37, Apr., 2013.

[116] N. Nigam and I. Kroo, *Persistent surveillance using multiple unmanned air vehicles*, in *IEEE Aerospace Conference*, (Big Sky, MT, USA), pp. 1–14, Mar., 2008.

[117] I. Maza and A. Ollero, *Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms*, in *Distributed Autonomous Robotic Systems 6* (R. Alami, R. Chatila, and H. Asama, eds.), pp. 221–230. Springer, 2007.

[118] J. Wood and J. K. Hedrick, *Space partitioning and classification for multi-target search and tracking by heterogeneous unmanned aerial system teams*, *Infotech@ Aerospace* (March, 2011).

[119] B. Bethke, M. Valenti, and J. P. How, *UAV task assignment*, *IEEE Robotics & Automation Magazine* **15** (2008), no. 1 39–44.

[120] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, *A survey of research on cloud robotics and automation*, *IEEE Transactions on Automation Science and Engineering* **12** (2015), no. 2 398–409.

[121] M. T. Hale and M. Egerstedt, *Differentially private cloud-based multi-agent optimization with constraints*, in *American Control Conference*, (Chicago, USA), pp. 1235–1240, July, 2015.

[122] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, *Control of multi-agent systems with event-triggered cloud access*, in *European Control Conference*, (Linz, Austria), pp. 954–961, July, 2015.

[123] K. Obermeyer, *Path planning for a uav performing reconnaissance of static ground targets in terrain*, in *AIAA Guidance, Navigation, and Control Conference*, (Chicago, USA), pp. 10–13, Aug., 2009.

[124] K. J. Obermeyer, P. Oberlin, and S. Darbha, *Sampling-based path planning for a visual reconnaissance UAV*, *AIAA Journal of Guidance, Control, and Dynamics* **35** (2012), no. 2 619–631.

[125] J. Isaacs and J. P. Hespanha, *Dubins traveling salesman problem with neighborhoods: A graph-based approach*, *Algorithms* **6** (2013), no. 1 84–99.

[126] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[127] P. Oberlin, S. Rathinam, and S. Darbha, *A transformation for a heterogeneous, multiple depot, multiple traveling salesmen problem*, in *American Control Conference*, (St. Louis, MO, USA), pp. 1292–1297, June, 2009.

[128] C. E. Noon and J. C. Bean, *A Lagrangian based approach for the asymmetric generalized traveling salesman problem*, *Operations Research* **39** (1991), no. 4 623–632.

[129] L. V. Snyder and M. S. Daskin, *A random-key genetic algorithm for the generalized traveling salesman problem*, *European Journal of Operational Research* **174** (2006), no. 1 38–53.

[130] R. T. Marler and J. S. Arora, *Survey of multi-objective optimization methods for engineering*, *Structural and multidisciplinary optimization* **26** (2004), no. 6 369–395.

[131] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, *Many-objective optimization: An engineering design perspective*, in *Evolutionary Multi-Criterion Optimization* (C. A. C. Coello, A. H. Aguirre, and E. Zitzler, eds.), pp. 14–32. Springer, 2005.

[132] D. Grundel and D. Jeffcoat, *Formulation and solution of the target visitation problem*, in *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, (Chicago, USA), Sept., 2004. AIAA 2004-6212.

[133] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1998.

[134] S. Ponda, H. Choi, and J. How, *Predictive planning for heterogeneous human-robot teams*, in *AIAA Infotech@ Aerospace*, (Atlanta, GA, USA), April, 2010. AIAA 2010-3349.

[135] C. Murray and W. Park, *Incorporating human factor considerations in unmanned aerial vehicle routing*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **43** (2013), no. 4 860–874.

[136] J. Peters, V. Srivastava, G. Taylor, A. Surana, M. P. Eckstein, and F. Bullo, *Mixed human-robot team surveillance: Integrating cognitive modeling with engineering design*, *IEEE Control Systems* **35** (2015), no. 6 57–80.

[137] J. Peters and L. Bertuccelli, *Robust task scheduling for multi-operator supervisory control missions*, *Journal of Aerospace Information Systems* (2016) 393–406.

[138] J. R. Peters, S. Wang, A. Surana, and F. Bullo, *Cloud-supported coverage control for persistent surveillance missions*, *ASME Journal of Dynamic Systems, Measurement, and Control* **139** (2017), no. 8 081011–081011–12.

[139] *User Manual: Tobii X60 and X120 Eye Trackers*, 2008. `http://www.tobii.com`.

[140] J. Slooter and D. Van Norren, *Visual acuity measured with pupil responses to checkerboard stimuli.*, *Investigative Ophthalmology & Visual Science* **19** (1980), no. 1 105–108.

[141] K. Savla and E. Frazzoli, *A dynamical queue approach to intelligent task management for human operators*, *Proceedings of the IEEE* **100** (2012), no. 3 672–686.

[142] M. Neerincx, *Cognitive task load design: Model, methods and examples*, *Handbook of cognitive task design* (2003) 283–305.

[143] Y.-F. Tsai, E. Viirre, C. Strychacz, B. Chase, and T.-P. Jung, *Task performance and eye activity: Predicting behavior relating to cognitive workload*, *Aviation, Space, and Environmental Medicine* **78** (2007), no. Supplement 1 B176–B185.

[144] J. Peters and L. Bertuccelli, *Robust scheduling strategies for collaborative human-uav missions*, in *American Control Conference (ACC), 2016*, (Boston, MA, USA), pp. 5255–5262, IEEE, July, 2016.

[145] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6. Athena Scientific, Belmont, MA, 1997.

[146] E. Lawler and D. Wood, *Branch-and-bound methods: A survey*, *Operations Research* **14** (1966), no. 4 699–719.

[147] T. Achterberg, T. Berthold, and G. Hendel, *Rounding and propagation heuristics for mixed integer programming*, in *Operations Research Proceedings 2011*, pp. 71–76. Springer, 2012.

[148] MathWorks, "Linear programming and mixed-integer linear programming." `http://www.mathworks.com/help/optim/linear-programming-and-mixed-integer-linear-programming.html`. Accessed: 2015-08-29.

[149] A. Makhorin, "GNU linear programming kit." `https://www.gnu.org/software/glpk/`, 2000–2012. Accessed 2017-02-22.

[150] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in *Recent Advances in Learning and Control* (V. Blondel, S. Boyd, and H. Kimura, eds.), Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[151] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." `http://cvxr.com/cvx`, Mar., 2014.

[152] D. N. Southern, *Human-guided management of collaborating unmanned vehicles in degraded communication environments*, Master's thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May, 2010.

[153] C. E. Nehme, *Modeling human supervisory control in heterogeneous unmanned vehicle systems*. PhD thesis, Massachussetts Institute of Technology, 2009.

[154] L. E. Dubins, *On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents*, *American Journal of Mathematics* **79** (1957) 497–516.

[155] U. Army, *Attack reconnaissance helicopter operations*, Tech. Rep. FM 3-04.126, Department of the Army (US), Feb., 2007. Retrieved from http://usacac.army.mil/sites/default/files/misc/doctrine/CDG/fms.html on Dec. 5, 2016.

[156] ILOG, Mountain View, CA, *ILOG CPLEX User's guide*, 1999.

[157] MathWorks, "Linear programming and mixed-integer linear programming." `http://www.mathworks.com/help/optim/linear-programming-and-mixed-integer-linear-programming.html`. Accessed: 2015-08-29.

[158] A. Ben-Tal, L. El-Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton University Press, 2009.

[159] G. C. Calafiore and M. C. Campi, *The scenario approach to robust control design*, *IEEE Transactions on Automatic Control* **51** (2006), no. 5 742–753.

[160] P. Kouvelis, R. Daniels, and G. Vairaktarakis, *Robust scheduling of a two-machine flow shop with uncertain processing times*, *IIE Transactions* **32** (2000), no. 5 421–432.